A decision support framework towards a simulation model for the risk-constrained vehicle routing problem

Nita-Maré Oosthuizen

Thesis presented in partial fulfilment of the requirements for the degree of
Master of (Industrial) Engineering
in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr DP Lötter

April 2019

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2019

Abstract

The vehicle routing problem is a well-researched problem in the operations research literature that typically involves the delivery of commodities to customers in a transportation network. One particular aspect thereof, however, that has not received as much attention is the security associated with the transportation of valuable commodities.

Customers typically rely on companies within the cash-in-transit industry to transport valuable goods between various locations. Due to the high value of these transported goods, the vehicles are constantly exposed to large amounts of risk and the entire cash-in-transit process is therefore susceptible to crime.

In this thesis, a *decision support system* (DSS) framework is put forward for mitigating risk along cash-in-transit routes. The risk along these routes is directly proportional to the amount of valuable goods on board a vehicle and the distance travelled by the vehicle. The proposed routes should, therefore, minimise the risk of the routes and facilitate effective trade-offs between a variety of decision criteria that are configurable by a user. Furthermore, it is envisioned that the DSS framework may be used as a basis for implementing a simulation model in the future.

The DSS contains three main components, namely a model base, a database and a user interface. The model base forms the heart of the DSS and is responsible for housing two algorithms that are both based on the Clarke-Wright savings algorithm. The first algorithm is for solving the capacitated vehicle routing problem, is based directly on the original Clarke-Wright algorithm, and does not include any risk constraint, while the second algorithm is for solving a risk-constrained variation on the vehicle routing problem and is based on a modification of the Clarke-Wright algorithm. The risk constraint adopted in this thesis involves the specification of a risk threshold to which each route must adhere. Furthermore, the user is able to provide specific input and parameter information through the user interface so as to configure constraints in order to customise the problem according to his or her preference. After a problem instance has been solved, the user interface provides a visual output to the user, which the user may analyse to draw certain conclusions.

The DSS is tested and validated using two different methods. The first method is a trace validation and the second method is a sensitivity analysis. The sensitivity analysis is performed in two parts, with the first part being based on test data from a benchmark library in the vehicle routing literature and the second part involves parameter variation. The results obtained during these tests are analysed, confirming that the DSS functions correctly.

The DSS is implemented in a modelling environment capable of solving both the capacitated vehicle routing problem and its risk-constrained counterpart. The system may be used to observe the effect of varying the risk threshold associated with a set of vehicle routes. It is found that the DSS produces good results when compared with results obtained from benchmark problems in the literature.

Uittreksel

Die voertuigroeteringprobleem is 'n goed-nagevorsde probleem in die operasionele navorsingsliteratuur en behels tipies die aflewering van kommoditeite tussen kliënte in 'n vervoernetwerk. Een spesifieke aspek daarvan wat egter nog nie soveel aandag gekry het nie, is die veiligheid wat verband hou met die vervoer van waardevolle kommoditeite.

Kliënte maak gewoonlik op maatskappye in die transito-bedryf staat om waardevolle goedere tussen verskillende plekke te vervoer. As gevolg van die hoë waarde van hierdie goedere word voertuie voortdurend aan groot risiko blootgestel en is die hele transito-proses dus vatbaar vir misdaad.

In hierdie tesis word 'n *besluitsteunstelsel* (BSS) raamwerk vir die verlaging van risiko langs kontant-in-transito-roetes voorgestel. Die risiko langs hierdie roetes is direk eweredig aan die hoeveelheid waardevolle goedere aanboord 'n voertuig en die afstand wat die voertuig aflê. Die voorgestelde roetes moet dus die risiko van die roetes verminder en 'n doeltreffende afruiling tussen 'n verskeidenheid besluitnemingskriteria wat deur 'n gebruiker konfigureerbaar is, fasiliteer. Daarbenewens word daar voorsien dat die BSS raamwerk as basis kan dien vir die toekomstige implementering van 'n simulasiemodel.

Die BSS bevat drie hoofkomponente, naamlik 'n modelbasis, 'n databasis en 'n gebruikerskoppelvlak. Die modelbasis vorm die hart van die BSS en bevat implementasies van twee algoritmes wat op die Clarke-Wright spaaralgoritme gebaseer is. Die eerste algoritme kan ingespan word vir die oplossing van die gekapasiteerde voertuigroeteringprobleem, is direk op die oorspronklike Clarke-Wright-algoritme gebaseer, en bevat geen risikobeperking nie, terwyl die tweede algoritme gebruik kan word om 'n risiko-beperkte variasie op die voertuigroeteringsprobleem op te los, en gebaseer is op 'n wysiging van die Clarke-Wright-algoritme. Die risikobeperking wat in hierdie tesis oorweeg word, behels die spesifikasie van 'n risikodrempel waaraan elke roete moet voldoen. Verder kan die gebruiker spesifieke inset- en parameterinligting deur middel van die gebruikerskoppelvlak verskaf om beperkings daar te stel en die probleem volgens sy of haar voorkeur aan te pas. Nadat 'n probleemspesifikasie opgelos is, bied die gebruikerskoppelvlak 'n visuele uitset daarvan aan die gebruiker, wat die gebruiker dan kan analiseer om sekere gevolgtrekkings te maak.

Die BSS word deur middel van twee verskillende metodes getoets en gevalideer. Die eerste metode is 'n spoorvalidering terwyl die tweede metode 'n sensitiwiteitsanalise is. Die sensitiwiteitsanalise word in twee dele uitgevoer. Die eerste gedeelte is gebaseer op toetsdata afkomstig van 'n maatstafbiblioteek in die voertuigroeteringsliteratuur en die tweede deel behels parametervariasie. Die resultate wat tydens hierdie toetse behaal word, word ontleed en bevestig dat die DSS korrek funksioneer.

Die BSS word in 'n modelleringsomgewing geïmplementeer wat in staat is om beide die gekapasiteerde voertuigroeteringsprobleem en sy risikobeperkte eweknie op te los. Die stelsel kan

gebruik word om die effek van wisselvalligheid in die risikodrempel wat met 'n stel voertuigroetes gepaard gaan, waar te neem. Daar word bevind dat die BSS goeie resultate lewer in vergelyking met die resultate wat vir toetsprobleme in die literatuur verkry word.

Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- My amazing God, for providing me with the strength, perseverance and discipline to face every challenge that crossed my path, but also for every opportunity and life lesson I have been blessed with during this two year journey. He showed me once again that all things are possible through Him.
- My supervisor during my masters studies, Dr Danie Lötter, for the opportunity to complete my masters degree. I also thank him for taking the time to read and review the written work in this thesis and for the opportunity he gave me to present my work at two ORSSA conferences during the course of this degree.
- Dr Brian van Vuuren, for introducing me to the AnyLogic modelling software and for helping me to overcome some of the steep learning curves of the software. I also thank him for the time he invested in me during my first year of masters studies.
- My friends and colleagues in the *Stellenbosch Unit for Operations Research and Engineering* (SUnORE) research group who have supported me and provided me with many good times, laughter and many opportunities throughout the past two years. They made the late nights and long days that were spent writing this thesis in the office much less lonely and much more enjoyable.
- My other friends and my family, for their unending love and ongoing support through both the good times and the difficult times. In particular I would love to thank my parents, Marius and Susan Oosthuizen, for providing me with endless amounts of encouragement, love and also the financial support to finish my masters degree in industrial engineering. I thank them for the amazing example they set for me every day. Also my brother and my sister, Johan and Suné Oosthuizen, for their love and for always supporting and encouraging me.

Table of Contents

List of Reserved Symbols

Vehicle routing problems in general

Symbol	Meaning
$\mathcal{A}$	A set of edges
$\mathcal{C}$	A set of customer nodes in a network
c_{ij}	The cost of travelling from node i to node j
$\delta(\mathcal{C})$	The cut-set of a subset $\mathcal{C}$
$D_{\max}$	The maximum allowable length of a route
d_j	The demand of customer j
$\mathcal{E}$	A set of undirected edges
$\mathcal{E}_{\mathrm{e}}$	The number of times an edge e is traversed
$\gamma(\mathcal{C})$	The minimum number of vehicles required to service all customers in set $\mathcal{C}$
G	A graph representing the customer network of nodes and edges
$\mathcal{K}$	A set of vehicles
m_e	The distance of edge e
Q	The capacity limit of all vehicles
q_k	The capacity of vehicle k
S	A subset of set $\mathcal{V}$
u_i	The volume of goods left over in a vehicle after servicing node i
$\mathcal{V}$	A set of all nodes in a network
x_{ij}	The decision variable taking the value 1 if edge (i, j) is traversed, or the value 0 otherwise
x_{ijk}	The decision variable taking the value 1 if edge (i, j) is traversed by vehicle k, or the value 0 otherwise

Risk constrained vehicle routing problem

Symbol	Meaning
β_i	The minimum allowable arrival time of a vehicle at customer i
C_{event}	The consequences of an unwanted event
δ	The maximum allowable similarity between space and time
D_i	The probability of loss of valuable goods if an unwanted event occurs
$\mathcal{N}$	A set of nodes
$\mathcal{P}_t$	A set of planned routes and schedules for day t
p_{event}	The probability that an unwanted event occurs
p_{ij}	The probability that an unwanted event is successful on the route between customer i and customer j
R_{event}	The risk of an unwanted event
R_i	The risk index at the previous customer in the route

R_j	The risk index at the current customer in the route
$\mathcal{T}$	A set of days
T	The risk threshold for a vehicle route

Clarke-Wright savings algorithm

Symbol	Meaning
$\overline{d}$	The average demand
λ	A parameter that attempts to avoid circular-shaped routes
μ	A parameter that attempts to exploit the asymmetry of the distance between two customers
s_{ij}	The savings achieved by linking customer i and customer j into a single route
v	A parameter that is used to regulate the magnitude of the effect of the demand size

List of Acronyms

ATM: Automated Teller Machine

CIT: Cash-In-Transit

CSV: Comma Separated Value

CVRP: Capacitated Vehicle Routing Problem

DSS: Decision Support System

DVRP: Dynamic Vehicle Routing Problem

ERD: Entity-Relationship Diagrams

FCFS: First Come First Serve

GIS: Geographical Information Systems

GUI: Graphical User Interfaces

HCI: Human-Computer Interaction

PCTVRP: Precedence Constrained Vehicle Routing Problem

RCVRP: Risk Constrained Vehicle Routing Problem

RCTVRP: Risk Constrained Cash-In-Transit Vehicle Routing Problem

SQL: Structured Query Language

TSP: Travelling Salesman Problem

VRP: Vehicle Routing Problem

VRPTW: Vehicle Routing Problem with Time-Windows

XML: Extensible Markup Language

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Contents

1.1 Background

The *vehicle routing problem* (VRP) has become a well distinguished problem among researchers and is commonly used in a wide variety of fields, typically with the purpose of moving commodities between customers in a network. The VRP has evolved over the years to encompass different variations of the problem that are applicable to a wide range of scenarios. Typical scenarios include the consideration of *deliveries and collections*, *time-window* variations and *dynamic* routing. One particular field of interest that has not received as much attention as of yet, however, is the *security and safety* associated with the transportation of valuable commodities between various customers or companies, such as banks, jewellers, casinos, ATMs, shopping centres and other large retailers.

Companies such as these listed above, typically rely on external companies within the *cash-in-transit* (CIT) industry in order to transport their valuable goods from their suppliers or to their customers, which typically includes commodities such as bank notes, coins, gold and other valuable items. Due to the value associated with these goods transported, CIT firms are constantly exposed to large amounts of risk and the entire CIT process becomes very susceptible to crime activities such as armed robberies, CIT vehicle heists and in some cases fatal shooting incidents. Even though the CIT industry has attempted to avoid or mitigate risk by making use of armoured vehicles as shown in Figure 1.1 that are equipped with vehicle-tracking devices, armed security, inter-locking doors and technologically advanced safes that all attempt to scare-off criminals or obstruct access to the valuable goods — the valuables carried by the CIT vehicle remain a soft target for criminals.

Data form the *South African banking risk information centre* (Sabric) [95] has shown an increase of 43% in the year 2016's CIT heists within South Africa alone. Furthermore, one of South

FIGURE 1.1: *An example of a CIT armoured vehicle. Images adopted from INKAS [65].*

Africa's local news platforms, News24 [97], reported that a prominent series of at least 140 CIT heists have occurred since January 2018, which equates to almost one heist per day for the first half of the year. Moreover, on a global scale according to the British Security Industry Association [20], 300kg of gold was stolen in Foggia (Italy) in the year 2013, while during the same time 50 million Euro was stolen in Brussels, and in the same year 30 million Euro was stolen in Varese (Italy). Furthermore, countries such as the UK are estimated to transport up to 500 billion Great British Pounds on a yearly basis, however, in the past they have also experienced losses of up to 150 million Great British Pounds due to CIT heists [20].

Figure 1.2 shows the amount of notes and coins present in South Africa's cash circulation cycle, as well as the growth thereof for the period January 2010 to January 2017. Note and coin usage has doubled from the year 2010 to 2017, and a further estimated growth increase of 6% is predicted by the Protea Coin Group [47] for cash circulation through to the year 2020. Therefore, as long as note and coin presence remains an integral part of world-wide circulation, it is vital to address the security and safety risks associated with the CIT industry [25, 95]. The various costs that are associated with cash tend to range far wider than merely the handling

FIGURE 1.2: *The amount of notes and coins that form part of the cash circulation in South Africa, as adapted from Maynard [90].*

costs. It also includes insurance costs, security costs, risk costs, transportation costs and holding costs [95]. The insurance costs associated with cash have, however, increased multiple times as the number of crime incidents related to the CIT industry have multiplied.

Criminals, that are generally very heavily armed, have become very well educated on the working of the CIT process, so much so that their success rate has lead to a subsequent 74% fatality rate of CIT officers and perpetrators, while the number of reported injuries related to CIT heists have increased by 32% over merely 10 months according to research presented by Moneyweb [95]. Smith and Louis [121] suggests that the primary reason for the increase in CIT heists is due to a lack of security analysis during the route planning phase of CIT operations and, therefore, more comprehensive planning is required in this phase..

FIGURE 1.3: *Examples of CIT heists and the damage and consequences that follow. Images adopted from the Sunday Times [89] and IOL [117], respectively.*

As mentioned previously, risk in the context of CIT vehicle routing has not received much attention in the VRP literature. A similar transportation problem, however, for the hazardous materials (*hazmat*) industry has been researched. For the transportation of hazardous materials, each road section is assigned a risk function and routes are selected so as to minimise operational costs. *The Center for Chemical Process Safety* (CCPS) [27] defines risk in the context of hazmat as an index of potential economic loss, environmental disturbance or damage, and human loss or injury that is quantified in terms of the probability that such an unwanted incident occurs. Furthermore, the Business dictionary [24] defines risk as the probability of damage, liability, suffering, injury or other inconveniences that happen as a result of internal or external susceptibilities, which may be altogether avoided if preventative action is taken. In the context of a hazmat transportation problem, an unwanted event would typically be an event where hazardous materials are exposed to humanity and, therefore holds horrific consequences on the surrounding area and people involved should an accident occur. In the CIT industry, however, the commodities that are transported are not hazardous or dangerous to humanity, yet they are of high value to the economy and, therefore the sabotage of these commodities also have severe consequences that have a devastating effect on the companies involved, and ultimately also effects the economy. Research has shown that the risk in the context of CIT routing is proportional to the route length, as well as the amount of valuable goods on-board the vehicle, and also that there are more risk involved for robberies and heists earlier in the day than later in the day [120].

There typically exists two types of undesirable consequences, namely *foreseeable consequences* and *unforeseeable consequences*. Foreseeable consequences are for the most part associated with the expected outcomes such as the anticipated loss of cash or other valuables when a CIT heist occurs. Unforeseeable consequences, on the other hand, are typically associated with events

and effects that cannot be anticipated beforehand, such as the criminal nature of the CIT heist and its subsequent effects such as loss or injury of CIT crew, damage to vehicles or equipment, medical treatments, and in some cases damage to road infrastructure. Although the total effect on the parties involved may be mitigated by the use of effective insurance for both types of consequences, there is no consolation to human loss or trauma [126]. It is, therefore, encouraged that preventative action is taken in order to mitigate of the probability that such events may occur or to take preventative action to reduce the impact of such an event, should it occur.

1.2 Informal problem description

The aim in this thesis is to develop a DSS framework towards a generic simulation model that is capable of suggesting suitable vehicle routes in order to mitigate the risk for CIT vehicle in the CIT industry, which they are exposed to while they travel along their vehicle routes. These routes should minimise the overall risk of the routes and embody effective trade-offs between a variety of decision criteria that is configurable by the user. The working of the model should be based on a mathematical modelling framework and decision support system that is designed with reference to the operations research literature pertaining to VRPs and the operations involved in the CIT industry. Furthermore, it is envisioned that the DSS framework may be used as a basis for implementing a simulation model in the future.

1.3 Thesis objectives

In order to effectively and efficiently model the anticipated risk associated with CIT vehicles, the following seven objectives will be pursued in this thesis:

I To *conduct* a comprehensive survey of the literature related to this thesis, which includes:

- (a) a detailed insight on the operation of CIT systems in the industry,
- (b) simulation modelling techniques and approaches,
- (c) VRPs in general,
- (d) VRPs in the context of risk mitigation,
- (e) DSSs in general,
- (f) data containing information on VRPs in the literature,
- (g) data containing information on CIT systems, and
- (h) additional information necessary to build a qualified model.

II To *design* a decision support framework that works toward a simulation model that is able to accurately prescribe CIT vehicles to routes that service customers effectively and efficiently while attempting to mitigate risk along all the various routes.

III The framework discussed in objective II requires:

- (a) a database that provides relevant data to the problem,
- (b) a hidden platform where the agents in the model work and interact,
- (c) an implementation of a CIT vehicle routing system,

(d) the minimisation of cost elements required to operate a CIT system (*i.e.* insurance cost, transportation cost and risk mitigation costs as suggested by Adendorff *et al.* [2]), and

(e) a vehicle routing model that is capable of calculating and suggesting suitable vehicle routes for CIT vehicles.

IV To *implement* Objective II along with a suitable *graphical user interface* (GUI) that enables the user to observe and interpret the process used to solve the problem. The user should be able to define parameters through the use of the GUI, as well as configure some constraints. The GUI should also display meaningful results to the user that (s)he can then interpret and use to make further decisions.

V To *validate* the model and GUI implemented in Objective IV, by testing the problem on benchmark problems from the VRP library, as well as benchmark problems relevant to the risk-constrained VRP. Furthermore, to ensure that Objective IV complies with all the requirements stated in Objectives I–III.

VI To *facilitate* various scenarios of the problem by making the model configurable to different scenarios (*i.e.* the model should be configurable to accommodate more than a single problem instance).

VII To *recommend* improvements for potential future work or additional features and enhancements that may be introduced to the model embodied in Objective IV.

1.4 Scope of the thesis

Due to the complex nature of CIT vehicle routing operations in the real-world and the multiple notions and concepts associated with the CIT process, the complexity that is involved in the proposed DSS model can easily become to large to incorporate within the time frame available for this thesis. It is, therefore, necessary to simplify the problem for the objectives, as well as the time constraints associated with this thesis. The thesis is, therefore limited to the some assumptions and these assumptions are discussed below.

This thesis will only focus on the CIT vehicle routing system as a whole and will, therefore, not consider individual client needs in terms of cash pick-up and deliveries for the various client types such as banks, jewellers, casinos, ATMs, shopping centres and other large retailers. It is assumed that the CIT vehicle is only responsible for *collecting* valuable goods along its route and not delivering any cash or valuables to other customers along the route. Thus, the vehicle leaves the depot in an empty state and is filled up along the route.

It is also assumed that a robbery may only occur *once* along a single route and, thus, the probability of being robbed more than once on a single route may be neglected.

Moreover, it is assumed that the robbers do not have an accomplice in the CIT vehicle and, therefore, they will not have any knowledge regarding the value of the goods being transported in any vehicle (*i.e.* the model does not account for corrupt companies or institutions). On longer routes it may, however, be assumed that robbers have more opportunities to rob the vehicle and that vehicles carrying more valuable goods accumulate a larger risk along its route since it has more valuables to lose. The vulnerability of each route (*i.e.* the probability that a robbery succeeds when it occurs) is assumed to be constant for all routes under consideration and is calculated according to historical data records of successful attacks as viewed from the

criminals' perspective [126] and, therefore, the vulnerability constant may be omitted in its entirety.

Due to security measures that are in place to access historical data related to CIT vehicle routing, the data required to build a realistic simulation model is not available. The focus of this thesis, therefore, shifts to build a DSS model in an environment that is able to accommodate simulation modelling in the future, or as the relevant data becomes available.

Winston [147] states that individuals' attitudes towards risk differ and that some individuals may be classified as more *risk inclined*, while others are classified as more *risk averse*, while some remain *risk neutral*. Risk inclined individuals seek risky opportunities, while risk averse individuals seek opportunities that present less risk. For the purpose of simplicity and continuity it is, however, assumed that all users that interact with the model in this thesis are risk-neutral in their decision making and expectations towards the model.

1.5 Research methodology

The methodology used to execute the research of this thesis is described below with reference to the objectives listed and discussed in §1.3.

First the author will *review* existing literature and *consult* existing companies, to understand how CIT systems work, as well as, what varieties of these system applications are available, and also how these systems are currently being implemented and used in practice. Next, the author will *acquire* the necessary skills to develop a DSS for a vehicle routing model using ANYLOGIC modelling software, after which the author will go forth to *develop* a DSS model that represents a network where CIT vehicles follow the best suited routes in order to service customers in a specified network while mitigating risk. Furthermore, the author will *develop* a GUI that enables the user to observe the execution of the DSS model and also enables the user to analyse the results which are finally displayed through the GUI, in order for the user to observe and draw his or her own conclusions. Next, the author will *implement* a suitable model validation by comparing the achieved results to benchmark data from the VRP online library [63], which will *ensure* that the developed DSS model is applicable to various scenarios (*i.e.* a generic model) and is also configurable by the user in order to accommodate user-specific problems. Finally, the author will *accommodate* and *include* recommended improvements and enhancements that may be added to the model, however, these changes and enhancements will only be implemented if they are currently vital to improve the model's quality, otherwise the author will advice these enhancements as future work.

1.6 Thesis organisation

Apart from this introductory chapter, this thesis contains six chapters. Each of these chapters are briefly summarised in this section.

Chapter 1 contains a brief background on CIT vehicle operating procedures and how these vehicles are used to transport valuable goods. The chapter provides some statistics on the relevance of cash circulation in South Africa, as well as in a global context. A discussion follows, which highlights the risks involved in the transportation of valuable goods in fulfilment of Objective I (a) of §1.3. A brief overview was given of other approaches that have been taken to combat risk in the context of CIT vehicle routing. A discussion followed on the thesis

problem statement, the seven objectives, the thesis scope and the research methodology followed throughout the project.

A comprehensive literature review is provided in Chapter 2 of the operations research literature pertaining to the VRP and its solution approaches in fulfilment of Objective I (c), (d), (f) and (g) of §1.3. The review covers literature on different formulations of the VRP, including the travelling salesman problem, the capacity constrained VRP, the distance constrained VRP, the time window constrained VRP, the precedence constrained VRP, the risk constrained VRP and the dynamic VRP. Furthermore, typical solution approaches that may be used to effectively solve the VRP are discussed and include the Clarke-Wright savings algorithm, the simulated annealing algorithm, the tabu search algorithm, the genetic search algorithm and the ant colony optimisation algorithm.

Furthermore, in Chapter 3 a brief review of the operations research literature pertaining to simulation modelling is provided in fulfilment of Objective I (b) of §1.3. The review covers literature on the history of simulation and the evolution thereof. The review also includes a discussion on the various simulation paradigms which include static versus dynamic models, deterministic versus stochastic models and continuous versus discrete models. In addition, the different levels of abstraction (*i.e.* low, medium and high abstraction) involved in the development of a simulation model were highlighted, as well as which level of abstraction is typically related to the different types of simulation models, including discrete event simulation modelling, agent-based simulation modelling, system dynamics simulation modelling and dynamic systems simulation modelling are also highlighted. Different elements involved in a simulation model and the advantages and disadvantages associated with simulation modelling are also mentioned. Next, the steps involved in the development of a simulation model are discussed in order to provide a sufficient understanding for the development of a DSS towards a simulation model and, finally, there is a discussion on the difference between optimisation and simulation.

Next, in Chapter 4 a literature review pertaining to DSSs are given in fulfilment of Objective I (e) and (h) of §1.3. The review covers literature on frameworks, with specific focus on decision support frameworks, the structure, the integrated components and system software that are encompassed in a DSS. Typical systems development methodology approaches including the waterfall methodology, the agile methodology and the object-oriented methodology are also discussed. Furthermore, research is conducted on the notion of testing, training and maintaining the system, which includes discussions on quality assurance, various testing methods, training techniques, system implementation and system maintenance methods.

Chapter 5 is devoted to a description of the developed DSS framework in fulfilment of Objectives II–IV and VI of §1.3. First, the modelling software that is used to develop the DSS is discussed along with its features. The implemented DSS contains three main components namely the database, the user interface and the model base. The model base is responsible for housing two models namely the capacitated vehicle routing model and the risk-constrained vehicle routing model. A user is allowed to provide specific input and parameter information through the GUI and this information is then considered, processed and optimised according to the user's preferences. The results displayed by the DSS are displayed to the user again through a GUI. The chapter provides the algorithm frameworks for each of the two models used in the model base. Furthermore, the implementation in the modelling environment is discussed by first defining the object classes involved in the model. Next, the GUI is described in the context of the modelling environment, as well as how the user is able to initialise the model through the use of the GUI. Finally, the model optimisation and visualisation are discussed in the context of the modelling environment.

Chapter 6 is devoted to testing and validating the working of the DSS proposed in Chapter 5 in fulfilment of Objective V of §1.3. First, the methods used to verify the working of the model are described in order to confirm that the model does indeed work according to the user-specified requirements. Furthermore, the model is validated using two techniques that were previously discussed in Chapter 4 including trace validation and a sensitivity analysis, in order to confirm that the model works correctly. The sensitivity analysis is performed in two parts, where the first part uses test data from the benchmark VRP library to validate the working of the capacitated vehicle routing model. The second part is a parameter variation that is implemented on the risk-constrained vehicle routing model in order to validate that it produces the correct results.

Chapter 7 is devoted to a conclusion of the work presented in the preceding chapters of the thesis. A summary of the chapters that are included in the thesis is provided, which is followed by a list of the contributions of the thesis and some possible avenues for expansion for future work on the research conducted in this thesis are given in fulfilment of Objective VII of §1.3.

Part I

Literature review

CHAPTER 2

Vehicle routing literature study

Contents

The material discussed in this chapter, provides a foundation for the understanding of VRPs from the operations research literature. The VRP has been researched by many researchers since its inception in 1959 and the evolution of VRPs is discussed and summarised in §2.1. The first, and most basic formulation of the VRP is the TSP and is discussed in §2.1.1. Many variations of the VRP model have been formulated over the years and typically include additional constraints such as capacity, distance, time, risk and dynamic constraints. These variations of the VRP model are discussed in §2.1.2–§2.1.7, respectively. The VRP is known as a combinatorial optimisation problem and, therefore, the use of heuristics or meta heuristics are recommended when attempting to solve such a problem. Some of the more popular solving approaches that may be used to solve the VRP are named in §2.2 and include the Clarke-Wright savings algorithm, the simulated annealing algorithm, the tabu search algorithm, the genetic algorithm, and the ant colony algorithm. Each of these solution approaches are discussed in sections §2.2.1–§2.2.5, respectively. The chapter finally closes in §2.3 with a brief review of the chapter contents.

2.1 A brief history on vehicle routing problems

The VRP is a well-known combinatorial optimisation problem in the operations research literature and was first introduced by Dantzig and Ramser [34] in which they followed a linear programming approach to formulate the first VRP. According to Toth and Vigo [136], the VRP may be defined as to find an optimal set of routes for a fleet of vehicles that depart from one or several depots to service (*i.e.* to typically deliver commodities) a set of customers (*i.e.* to typically deliver commodities) and to return back to the depot once they have finished serving their set of customers. In the VRP, each individual vehicle is required to start at a depot and return to the depot again once it has finished all its routes. Furthermore, a single vehicle may visit each customer at most once and each customer may only be serviced by a single vehicle.

The VRP is used to solve problems in many application areas such as supply chains, logistics, air-cargo, the weapon assignment problem and in the management of deliveries of goods and services. Numerous VRP models exist in the literature which aim to achieve a specific goal. The VRP is also flexible in the sense that it may be customised by adding various supplementary constraints. It may, therefore be altered according to user-specified requirements in order to make the problem more user-specific [108]. Supplementary constraints and requirements may be added depending on the nature of the goods transferred or carried, the quality of the service required, and the character traits of the customers and vehicles that are included in the problem formulation. These supplementary constraints may include vehicle-specific *load* restrictions, *distance* limitations on some of the vehicle routes, *time* constraints where delivery or pick-up of commodities have to take place within certain time windows, *vehicle types* (*i.e.* heterogeneous or homogeneous vehicles), *precedence* organisation between the customers that need to be serviced, unknown *customer demands* and the *number of vehicles* that are allowed to service a single customer [108].

		Information quality	
	Input	**Deterministic**	**Stochastic**
Information evolution	**Known**	Static and deterministic	Static and stochastic
	Changes	Dynamic and deterministic	Dynamic and stochastic

TABLE 2.1: *Four classes of VRPs, adapted from Pillac et al. [108].*

VRPs may be divided into four classes by considering four characteristics *i.e.* static characteristic, dynamic characteristic, stochastic characteristic and deterministic characteristic. The first class of VRPs contains the *static and deterministic* VRPs. In these types of problems, all inputs are *known in advance* and no changes will occur to vehicle routes once the problem is in execution — the *routes are static* and not dynamic. Furthermore, this class of VRP exhibits a *deterministic* characteristic, since the outputs of the system correspond with the inputs that were provided (*i.e.* sensible inputs will produce sensible outputs, and non-sensible inputs will produce non-sensible outputs). Authors which have contributed to formulations of the VRP in this class include Baldacci [7], Cordeau [31], Laporte [76], and Toth and Vigo [136].

The next class of VRPs comprise the *static and stochastic* VRPs. In these types of problems, some inputs are only *partially known* — typically as random (or *stochastic*) input variables.

Routes are also typically designed prior to the execution process and only minor changes occur after the problem is solved. Examples of random input variables included in this class include stochastic *customers* where a customer is serviced with a certain probability, stochastic *times* where travel times or service times are modelled as random variables, and stochastic *demands* where customer demands are introduced randomly. Authors which have contributed to formulations of the VRP in this class include Bertsimas and Simchi-Levi [11], and Cordeau *et al.* [31, 30].

The next class of VRPs comprise the *dynamic and deterministic* VRPs. In these types of problems, most of the inputs are unknown and are therefore determined dynamically while the routes are being calculated (*i.e.* the routes continue *changing* while the problem is being solved), while a small part of the input remains *deterministic*. This type of VRP requires real-time communication between the vehicles in the system and the decision maker, for example mobile device communications or the use of global positioning systems as cited by Jaillet and Wagner [66].

The final class of VRPs comprise the *dynamic and stochastic* VRPs. In these types of problems, most of the inputs are unknown and determined dynamically while the routes are being calculated. The difference between this class of VRP and the dynamic and deterministic class of VRP is that in the dynamic and deterministic class of VRP, existing knowledge is continually made available. Routes may therefore be redefined dynamically (for example to include a new customer on the route that instantaneously also requires service) and are not constrained to a definite final answer from the start of the problem execution. Examples where the dynamic VRP has been applied is in problems where customer services are required to be exactly sequenced along routes, managing fleets of emergency vehicles, and dynamic allocation problems as mentioned by Spivey and Powell [124].

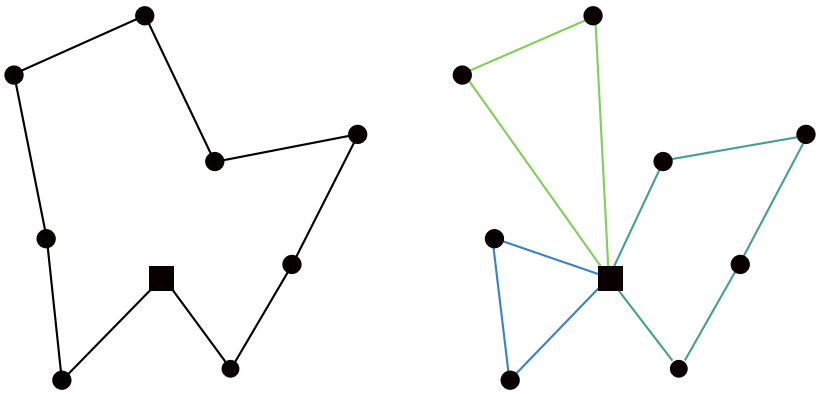
Since there exists no known polynomial algorithm that is able to find the optimal solution to each instance of the problem, the CVRP problem is known to be NP-hard [7, 136]. A NP-hard problem (abbreviated from *non-deterministic polynomial-time hard problem*) describes a problem where no polynomial-time exact solution techniques are known by which members of this class of problem may be solved.

The most basic form of the VRP is known as the *travelling salesman problem* (TSP), which involves finding an optimal set of routes for a single vehicle that services a set of customers. The TSP forms the basis on which VRPs in the operations research literature are formulated. The next section is therefore devoted to a discussion on the TSP and how complexity may be added to the TSP in order to arrive at more complex variations thereof *i.e.* formulations of the VRP.

2.1.1 The travelling salesman problem

In the TSP, there is a set of cities and one salesman, which has to find the cheapest route in order to service all the cities in the network (*i.e.* minimise the travel cost). The objective in the TSP is therefore to minimise the overall cost of travelling between the cities. The salesman starts and ends the trip from his/her home-town. Each city in the list is represented by a node and the routes connecting the nodes are called edges. These nodes and edges may be represented by a graph $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ denotes a set of m vertices (*i.e.* cities in the context of a TSP) and $\mathcal{A}$ denotes the edges (i, j) that connect the vertices (*i.e.* the possible routes connecting cities i and city j). The cost of travelling on an edge between two cities (*i.e.* from city i to city j) is denoted by c_{ij} [60]. In the case where all the vertices (cities) in the graph are interlinked (*i.e.* all vertices (cities) can be reached from all other vertices (cities) directly), the graph is referred to as a *complete* graph. When all of the vertices (cities) in the list are visited exactly

once by the salesman, the edges that were traversed during the trip are referred to as a *tour* or a *Hamiltonian cycle* [60]. A TSP involving a single vehicle that is used to traverse the edges is illustrated graphically in Figure 2.1(a), while a VRP involving more than one vehicle to visit the same set of vertices (cities) for a set of vertices (cities) is illustrated graphically in Figure 2.1(b).



(a) *TSP* (b) *VRP*

FIGURE 2.1: *The route for a single vehicle servicing a set of cities in a TSP in (a) and the routes for a number of vehicles servicing the same set of cities in a VRP in (b).*

A TSP may be defined as either *symmetric* or *asymmetric*. A symmetric TSP consists of edges that are the same distance from vertex (city) i to vertex (city) j and *vice versa*, whereas an asymmetric TSP consists of edges where the distance from vertex (city) i to vertex (city) j is different than the distance from vertex (city) j to vertex (city) i. The asymmetric TSP was formulated by Miller *et al.* [94] in 1960 and aims to minimise the overall travel costs of the routes. Let $\mathcal{C}$ denote a set of vertex (city) nodes $i \in \mathcal{C}$ where $i = 1, \ldots, m$, and let x_{ij} denote the decision variable taking the value 1 if edge (i, j) is traversed, or the value 0 otherwise. The objective in the asymmetric TSP is then to

$$\text{minimise} \sum_{j=1}^{m} \sum_{i=1}^{m} c_{ij} x_{ij}, \tag{2.1}$$

subject to the constraints

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1, \ldots, m, \tag{2.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, \ldots, m, \tag{2.3}$$

$$u_i - u_j + 1 \leq (m-1)(1 - x_{ij}), \qquad i, j = 2, \ldots, m, \tag{2.4}$$

$$2 \leq u_i \leq m, \qquad i = 2, \ldots, m, \tag{2.5}$$

$$u_1 = 1, \tag{2.6}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \mathcal{C}, \tag{2.7}$$

where u_i ensures that sub-tours are eliminated from the solution. Constraint set (2.2) ensures that the salesman may only travel from vertex (city) i to vertex (city) j once, while constraint set (2.3) ensures that the salesman may only travel from vertex (city) j to vertex (city) i exactly once. Constraint sets (2.4)–(2.6) ensures that any possible sub-tours are eliminated by using the so-called *Miller-Tucker-Zemlin* (MTZ) [94] sub-tour elimination formulation as formulated by

Pataki [106]. It may be observed that in constraint set (2.4) the arc-constraint forces $u_j \geq u_i + 1$ which results in $x_{ij} = 1$, whereas if $x_{ij} = 0$ the constraint is not binding. Finally, constraint set (2.7) ensures the binary nature of the decision variable.

In the symmetric TSP, the distance travelled along an edge is the same in both directions and, therefore, $c_{ij} = c_{ji}$ for all $i, j \in \mathcal{C}$. This may be substituted into the asymmetric TSP equations (2.1)–(2.7) in order to formulate a symmetric TSP. The symmetric TSP is a special instance of the asymmetric TSP, and it may therefore be assumed that the asymmetric TSP formulation may be used to solve the symmetric TSP [136]. It has, however, been found that the asymmetric TSP formulation performs badly when used to solve the symmetric TSP. The decision variable is therefore formulated differently in the symmetric TSP to improve the quality of the solution achieved. Let x_{ij} denote the decision variable taking the value 1 if edge (i, j) is traversed, or the value 0 otherwise. Furthermore, (i, j) denotes the set of edges in $(i, j) \in \mathcal{A}$. The objective in the symmetric TSP is then to

$$\text{minimise} \sum_{i<j}^{m} c_{ij} x_{ij} \tag{2.8}$$

subject to the constraints

$$\sum_{j=2}^{m} x_{1j} = 2, \tag{2.9}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \qquad k = 2, \ldots, m, \tag{2.10}$$

$$\sum_{i<j} x_{ij} \leq |\, S \,| - 1, \qquad i, j \in S, \tag{2.11}$$

$$3 \leq |\, S \,| \leq n - 2, \qquad S \subset \mathcal{V} \setminus \{1\}, \tag{2.12}$$

$$x_{ij} \in \{0, 1\}, \qquad 1 < i < j \tag{2.13}$$

$$x_{1j} \in \{0, 1, 2\}, \qquad j = 2, \ldots, m. \tag{2.14}$$

Constraint sets (2.9) and (2.10) are known as the degree constraints that specify that each vertex (city) may only be entered and left exactly once, while constraints sets (2.11) and (2.12) ensure that no sub-tours are formed. Hence, if a sub-tour existed on a subset S of the vertices (cities) in $\mathcal{V}$, then the sub-tour would contain $|\, S \,|$ arcs and the same number of vertices (cities). Constraint (2.12), therefore, constrains the value of $|\, S \,|$. Furthermore, since the problem is symmetric, only the upper triangular distance matrix should be considered, therefore, the variable x_{ij} is only defined for $i < j$, as shown in constraint set (2.13). It should be noted that the symmetric TSP (2.8)–(2.14) is not a pure binary integer formulation since the variable x_{1j} can be either $0, 1$ or 2, as shown in constraint set (2.14).

2.1.2 Capacity constrained VRP

A well-known variation of the VRP is the *capacitated vehicle routing problem* (CVRP). In the CVRP, vehicle routes are constrained by the capacity limitations of the vehicles that are utilised to transport the goods between customers. Furthermore, the assumption is made that all vehicles are homogeneous and that each vehicle starts from a common depot and returns to the same depot once it has finished its routes. Let k denote a vehicle used in the fleet. The objective in the CVRP is to minimise the total distance travelled by all vehicles, while simultaneously ensuring that customer demands are met. The problem is constrained in such a way that each

customer is visited exactly once and that the sum of all customer demands on a single route may not exceed the vehicle capacity.

The depot is located at vertex 0. Furthermore, let d_0 denote the demand for the depot having a value 0 for the sake of completeness. The remaining vertices are the customers that form part of the network and may be denoted as a subset $\mathcal{C}$ of $\mathcal{V}$, each having a demand d_j. Similar to the graph G that was explained in the previous section, each of the edges (i, j) that connect the vertices in $\mathcal{V}$ have a non-negative travel distance c_{ij} associated with it. Furthermore, similar to the TSP, the CVRP may be formulated symmetrically or asymmetrically. In a similar fashion as with the TSP, the distances in the asymmetric CVRP are directed edges ($c_{ij} \neq c_{ji}$), whereas the distances in the symmetric CVRP are undirected edges ($c_{ij} = c_{ji}$), for all edges $i, j \in \mathcal{V}$.

Let $\mathcal{C} = \mathcal{V} \setminus \{0\}$ denote the set of customers excluding the depot and let $\gamma(\mathcal{C})$ denote the minimum number of vehicles required to service all the customers in some subset of $\mathcal{C} \subseteq \mathcal{V}$. The $\gamma(\mathcal{C})$ parameter may be approximated by solving a *Knapsack Problem*[1]. Furthermore, let x_{ij} denote the decision variable taking the value 1 if the arc $(i, j) \in \mathcal{A}$ is traversed by some vehicle, or the value 0 otherwise. The objective in the asymmetric CVRP is then to

$$\text{minimise} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} x_{ij}, \tag{2.15}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, \ldots, m, \tag{2.16}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1, \ldots, m, \tag{2.17}$$

$$\sum_{i=1}^{m} x_{i1} = K, \tag{2.18}$$

$$\sum_{j=1}^{m} x_{1j} = K, \tag{2.19}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{m} x_{ij} \geq \gamma(\mathcal{C}), \qquad \mathcal{C} \subseteq \mathcal{V} \setminus \{0\},\ \mathcal{C} \neq \emptyset, \tag{2.20}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \mathcal{V}. \tag{2.21}$$

Constraint sets (2.16) and (2.17) ensure that exactly one vehicle services each customer once, while constraints (2.18) and (2.19) ensure that a vehicle departs from the depot and returns back to the depot once it has completed its route. Constraint set (2.20) is known as a capacity-cut constraint which ensures that enough vehicles are assigned to a route in order to satisfy the demand of all customers on the specified route. Finally, constraint set (2.21) ensures the binary nature of the decision variables.

Constraints (2.16)–(2.19) may be combined to formulate one constraint as

$$\sum_{i \notin \mathcal{C}} \sum_{j \in \mathcal{C}} x_{ij} - \sum_{i \in \mathcal{C}} \sum_{j \notin \mathcal{C}} x_{ij} = 0, \qquad \mathcal{C} \subseteq \mathcal{V} \setminus \{0\},\ \mathcal{C} \neq \emptyset, \tag{2.22}$$

[1] A knapsack problem is a problem where a subset of n items have to selected to be packed into a knapsack in order to maximise the corresponding profit value, without exceeding the knapsack's capacity limit.

which ensures that each vehicle that enters a vertex (customer) is also required to exit the vertex (customer). Constraints (2.16)–(2.19) may therefore be replaced by constraint (2.22). In a similar fashion constraint set (2.20) may be substituted in constraint set (2.22) in order to derive the constraint

$$\sum_{i\notin\mathcal{C}}\sum_{j\in\mathcal{C}} x_{ij} \geq \gamma(\mathcal{V}\setminus\mathcal{C}), \qquad \mathcal{C}\subset\mathcal{V}. \tag{2.23}$$

which ensures that constraint set (2.23) results in an exponential increase[2] in the size of problem since the number of customers considered increases [76]. This results in the model becoming almost unsolvable for very large problem instances. This problem may, however, be solved by relaxing the sub-tour constraints (2.22)–(2.23) and by only implementing them when there is indeed a sub-tour violation, rather than implementing constraint set (2.20) in the model [76]. This is illustrated in the TSP model formulation of Miller *et al.* [94] where it is shown if a vehicle travels directly from customer i to customer j or not. Therefore, constraint set (2.20) may be replaced with

$$u_i - u_j + Qx_{ij} \leq Q - d_j, \quad j\in\mathcal{V}\setminus\{0\},\ i\neq j \text{ and } d_i+d_j\leq Q, \tag{2.24}$$

$$d_i \leq u_i \leq Q, \quad i\in\mathcal{V}\setminus\{0\}, \tag{2.25}$$

where $u_i\in\mathcal{V}\setminus\{1\}$ denotes an auxiliary variable indicating the instantaneous volume of goods in the vehicle after having serviced customer $i\in\mathcal{V}$. Constraint sets (2.24)–(2.25) ensure that the capacity of each vehicle is not exceeded and also ensures that sub-tours are eliminated. Note that constraint set (2.24) is not binding when $x_{ij}=0$. If $x_{ij}=0$, then $u_i\leq Q$ and $u_j\geq d_j$ for all $i,j\in\mathcal{V}$. If, however, $x_{ij}=1$, constraint set (2.24) forces the inequality $u_j\geq u_i+d_j$ for all $i,j\in\mathcal{V}$.

Dessouky *et al.* [40] also added to the available VRP formulations in the literature and formulated the asymmetric CVRP model (2.16)–(2.21) in a so-called *three-index form*. The objective in this model is to minimise the travel cost incurred by the vehicle fleet. Let Q denote the capacity limit of a vehicle and let x_{ijk} denote the decision variable taking the value 1 if the arc $(i,j)\in\mathcal{A}$ is traversed by vehicle k, or the value 0 otherwise. Following the same notion as in the asymmetric CVRP (2.15)–(2.21), the three-indexed asymmetric VRP aims to

$$\text{minimise} \sum_{j\in\mathcal{V}}\sum_{i\in\mathcal{V}}\sum_{k\in\mathcal{K}} c_{ij}x_{ijk}, \tag{2.26}$$

subject to

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{V}} x_{ijk} \geq 1, \qquad j=1,\ldots,m, \tag{2.27}$$

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} d_j x_{ijk} \leq Q, \qquad k=1,\ldots,K, \tag{2.28}$$

$$\sum_{i\in\mathcal{V}} x_{1ik} - \sum_{j\in\mathcal{V}\setminus\{0\}} x_{j1k} = 0, \qquad k=1,\ldots,K, \tag{2.29}$$

$$\sum_{i\in\mathcal{V}} x_{ijk} - \sum_{\ell\in\mathcal{V}\setminus\{0\}} x_{j\ell k} = 0, \qquad k=1,\ldots,K,\ j=1,\ldots,m, \tag{2.30}$$

$$\sum_{j\in\mathcal{V}} x_{1jk} = 1, \qquad k=1,\ldots,K, \tag{2.31}$$

$$x_{ijk}\in\{0,1\}, \qquad i,j=1,\ldots,m,\ k=1,\ldots,K. \tag{2.32}$$

[2] An exponential increase in the size of problem refers to the number of distinct non-empty subsets that may be formed from a set of n distinguishable objects which is 2^{n-1} [7].

Constraint set (2.27) ensures that each customer in the set $\mathcal{V}$ is visited exactly once, while constraint set (2.28) ensures that the maximum allowable capacity of each vehicle is not exceeded. Constraint sets (2.29)–(2.31) ensure that each vehicle $k \in \mathcal{K}$ visits an unique set of customers, and that each route starts and ends at the common depot. Furthermore, constraint sets (2.29)–(2.32) ensure that sub-tours are eliminated [3]. Finally, constraint set (2.32) ensures the binary nature of the decision variables.

Pataki [106] provides an alternative formulation for the elimination of sub-tours by using the MTZ sub-tour elimination constraints. He adapted them in order to be used in the three-indexed CVRP formulation (2.26)–(2.32). The following constraints may be used to supplement the CVRP formulation.

$$u_1 = 1, \qquad k \in \mathcal{K}, \tag{2.33}$$

$$2 \leq u_i \leq n, \qquad i \in \mathcal{V} \setminus \{0\}, \tag{2.34}$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ijk}), \qquad i, j \in \mathcal{V} \setminus \{0\}. \tag{2.35}$$

In the formulations shown thus far, the directions in which vehicles traverse arcs are important because of their asymmetric nature of the routes. In the case where the direction of the edges are unimportant, the set of arcs $\mathcal{A}$ may be replaced with a set of undirected symmetrical edges denoted by $\mathcal{E}$.

Laporte *et al.* [76] proposed a formulation for the symmetric CVRP in which the aim is to minimise the distance travelled by each vehicle. Let ξ_e denote the number of times an edge is traversed by a vehicle and let $\delta(\mathcal{C})$ denote the cut-set of a subset $\mathcal{C}$ of vertices. The cut-set $\delta(\mathcal{C})$ of a set $\mathcal{C}$ refers to the set of all edges in G which disconnects the sub-graph formed by the vertices in $\mathcal{C}$ from the remainder of the graph when removed. Furthermore, let $\gamma(\mathcal{C})$ denote the minimum number of vehicles required to service a subset of customers in $\mathcal{C}$, while m_e denotes the distance of edge e. The objective in the symmetric CVRP is to

$$\text{minimise} \sum_{e \in \mathcal{E}} m_e \xi_e, \tag{2.36}$$

subject to

$$\sum_{e \in \delta(\{i\})} \xi_e = 2, \qquad i \in \mathcal{V} \setminus \{0\}, \tag{2.37}$$

$$\sum_{e \in \delta(\{i\})} \xi_e = 2K, \tag{2.38}$$

$$\sum_{e \in \delta(\mathcal{C})} \xi_e \geq 2\gamma(\mathcal{C}), \qquad \mathcal{C} \subseteq \mathcal{V} \setminus \{0\},\ \mathcal{C} \neq \emptyset, \tag{2.39}$$

$$\xi_e \in \{0, 1\}, \qquad e \notin \delta(\{0\}), \tag{2.40}$$

$$\xi_e \in \{0, 1, 2\}, \qquad e \in \delta(\{0\}). \tag{2.41}$$

Constraint set (2.37) ensures that each customer is serviced exactly once on a single route by a single vehicle, while constraint set (2.38) creates K routes and constraint set (2.39) ensures that a sufficient amount of vehicles are assigned to satisfy the total demand of all customers in any subset of customers in $\mathcal{C}$. Since the symmetric CVRP is NP-hard, the parameter $\gamma(\mathcal{C})$ is, therefore, NP-hard to compute, since it requires the approximation of a knapsack problem (as described previously) where the minimum number of vehicles required to service a given subset of customers are represented by $\gamma(\mathcal{C})$, each with a capacity of Q. Furthermore, the decision

variable (2.40) ensures that each edge may be traversed exactly once by a single vehicle. Finally, the decision variable (2.41) ensures that an entire route may be dedicated to serving only a single customer.

2.1.3 Distance constrained VRP

The next variant of the VRP is the *distance constrained* VRP [82]. The distance constrained VRP is similar to the CVRP, however, the difference is that a constraint which limits the route distance travelled by a single vehicle is added in the distance constrained VRP. Let D_{max} denote the maximum allowable length of a route. Following the same notion as in the CVRP model (2.36)–(2.41), the constraint set

$$\sum_{i=0}^{N} c_{ij} \sum_{i=0}^{N} x_{ijk} \leq D_{max}, \qquad j \in \{1, \ldots, N\},\ k \in \{1, \ldots, K\}, \tag{2.42}$$

is added to the formulation.

2.1.4 Time window constrained VRP

A significant contribution to the formulation of VRPs is the *vehicle routing problem with time windows* (VRPTW). The aim in the VRPTW is typically to minimise travel costs while serving a subset of customers within specific predefined time windows and, subsequently limiting the customer "dissatisfaction" associated with delivering goods to the customers on a late schedule.

Similar to the VRPs described in earlier sections, the VRPTW involves finding the composition set of routes for an optimal number of heterogeneous vehicle which depart from a common depot, in order to service a set of customers with known demands [32]. In the VRPTW, each customer is associated with a *time window* or service time frame during which a customer requires service. These time windows consist of a subset of feasible consecutive time stages during which a customer may be serviced [122]. Furthermore, each customer has *service time* associated with it which refers to the number of time stages a customer requires for service to be performed sufficiently (*e.g* for the loading and unloading of goods at a customer). The depot is associated with a *scheduling horizon*, which in essence, serves the same purpose as the time windows associated with each customer, however, the scheduling horizon is a representation of the total time frame within which customers may be serviced. Furthermore, a vehicle is allowed to service more than one customer on a single route, but only if the time windows of the respective customers on the route do not over lap. Finally, the accumulated demand of the customers on the route may not exceed the capacity limit of the vehicle.

Kumar and Panneerselvam [74] propose that an earliest arrival time and a latest arrival time be specified for each customer. This interval is called the the arrival interval and represents ideal during which a vehicle may arrive at a customer. If a vehicle arrives at a customer earlier than the earliest arrival time, a waiting time is incurred, which is considered as time wasted. If a vehicle arrives at a customer later than the latest arrival time, a penalty is incurred which is a representation of the "dissatisfaction" of a customer. Let f_i denote the service time of a customer i and let T_i denote the arival time of customer i and let w_i denote the waiting time of customer i. Furthermore, let x_{ijK} denote the decision variable taking the value 1 if the arc $(i, j) \in \mathcal{A}$ is traversed by vehicle k, or the value 0 otherwise. The depot is again located at vertex 0. In order to simplify the problem, all distances are represented as Euclidean distances and all vehicle speeds are assumed to be unity. Hence, the travel cost, travel time and Euclidean

distance are considered to be equal to each another. Furthermore, each vehicle route is required to start from and end at the common depot and each route is assigned a single vehicle. Each route in the network has cost c_{ij} associated with it and travel time t_{ij}, where t_{ij} denotes the time it takes to travel from customer i to customer j. Furthermore, each vehicle is assigned a capacity limit denoted as q_k, and each customer has a known demand denoted as m_i (where $m_i \leq q_k$). The objective in the VRPTW is therefore to

$$\text{minimise} \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{j\neq i,k=1}^{K} c_{ij} x_{ijk}, \tag{2.43}$$

subject to

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} \leq K, \qquad i = 0, \tag{2.44}$$

$$\sum_{j=1}^{N} x_{ijk} = 1, \qquad i = 0,\ k \in \{1, \ldots, K\}, \tag{2.45}$$

$$\sum_{j=1}^{N} x_{jik} = 1, \qquad i = 0,\ k \in \{1, \ldots, K\}, \tag{2.46}$$

$$\sum_{k=1}^{K} \sum_{j=0, j\neq i}^{N} x_{ijk} = 1, \qquad i \in \{1, \ldots, N\}, \tag{2.47}$$

$$\sum_{k=1}^{K} \sum_{i=0, i\neq j}^{N} x_{ijk} = 1, \qquad j \in \{1, \ldots, N\}, \tag{2.48}$$

$$\sum_{i=1}^{N} m_i \sum_{j=0, j\neq i}^{N} x_{ijk} \leq q_k, \qquad k \in \{1, \ldots, K\}, \tag{2.49}$$

$$\sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} x_{ijk}(t_{ij} + f_i + w_i) \leq r_k, \qquad k \in \{1, \ldots, K\}, \tag{2.50}$$

$$\sum_{k=1}^{K} \sum_{i=0, i\neq j}^{N} x_{ijk}(T_i + t_{ij} + f_i + w_i) \leq T_j, \qquad i \in \{1, \ldots, K\}, \tag{2.51}$$

$$e_i \leq (T_i + w_i) \leq l_i \qquad i \in \{1, \ldots, N\}, \tag{2.52}$$

$$T_0 = w_0 = f_0 = 0, \tag{2.53}$$

where r_k denotes the maximum route time allowed for a vehicle k, e_i denotes the earliest arrival time at node i, and l_i denotes the latest arrival time at node i. Constraint set (2.44) ensures that exactly K routes start from and end at the depot, while constraint set (2.45) ensures that a single vehicle is assigned to a route that starts from the depot. In addition, constraint set (2.46) ensures that a single vehicle is assigned to a route that ends at the depot. While, constraint sets (2.47)–(2.48) ensure that a single vehicle visits each customer exactly once (*i.e.* a single vehicle services each customer on a route). Furthermore, constraint set (2.49) ensures that the service demands of all customers on a single route does not exceed the maximum capacity of the vehicle servicing the route, while constraint set (2.50) ensures that the travel time of each route is less than or equal to the maximum allowable travel time of the route. Constraint set (2.51) ensures that the arrival time of a vehicle does not exceed the specified arrival time of the

customer, while constraint set (2.52) ensures that the arrival time and the waiting time spent at a customer exceeds or is equal to the earliest arrival time of the customer, but less than or equal to the latest arrival time of the customer. Constraint set (2.52) may therefore be seen as a time window constraint ensuring that a customer is serviced during its time window. Finally, constraint set (2.53) ensures that the arrival time, waiting time and service time of the depot is set to zero.

Two types of constraints, namely *hard* and *soft* constraints exist in optimisation problems [74]. A constraint is considered hard if it has to be satisfied, while a soft constraint may be violated in some way. If a soft constraint is typically violated, a penalty is incurred in the objective function in order to ensure that violation is kept to a minimum. In the context of the VRPTW the time window constraint (2.52) may typically be formulated as a soft constraint depending on how strict the constraint must be, and subsequently how large the penalty should be if a vehicle does not meet the time window requirements. In real-world scenarios, various aspects such as traffic and road infrastructure may have an significant effect on the punctuality of a vehicle. It may therefore be beneficial to model the time window constraint in a soft manner so as to provide flexibility in terms of allowing a vehicle to arrive a few minutes late.

The VRPTW also belongs to the class of NP-hard problems and it is therefore recommended by Lenstra and Kan [81] that such problems be solved using heuristic methods. Meta heuristics such as the tabu search [111, 125], simulated annealing [28], genetic algorithm [110, 134], evolutionary strategies [62], large neighbourhood search [118], guided local search [70] and greedy randomised search [72] and multiple ant colony systems [52] may be used to solve variants of the VRPTW.

2.1.5 Precedence constrained VRP

The next variant of the VRP is the *precedence constrained* VRP (PCVRP) which is a slight variation of the VRPTW model (2.44)–(2.53). In the PCVRP the objective in the PCVRP is to minimise the total distance travelled, however, a constraint is added which forces the precedence of customers served on a route (customer i must be served before customer j or *vice versa*). Let t_{ik} denotes the arrival time of vehicle k at customer i, and PT_{ij} denotes the temporal offset of the least amount of time units that customer j may be visited after customer i, thus enforcing the precedence of customers on a route. The precedented pairs of customers are stored in the set $\mathcal{P}^{\text{prec}}$, as defined and formulated by Bredström and Rönnqvist [19]. The objective is, therefore the same as in 2.43 and is subject to an additional constraint

$$\sum_{k\in\mathcal{K}} t_{ik} \leq PT_{ij} + \sum_{k\in\mathcal{K}} t_{jk} \qquad i,j \in \mathcal{P}^{\text{prec}}, \tag{2.54}$$

where constraint set (2.54) ensures that some customers are given precedence during route construction. The formulations also includes constraint sets (2.44)–(2.49) as formulated by Kumar and Panneerselvam [74] in the VRPTW.

2.1.6 Risk constrained VRP

Another formulation of the VRP is the *risk constrained vehicle routing problem* (RCVRP). The RCVRP is a recently new variant of the VRP and was introduced by Talarico *et al.* [126] in 2015 and has not been researched as extensively as the other formulations of the VRPs available in the literature. The RCVRP focus specifically on the safety of valuable goods during transportation.

A large number of organisations such as large retailers, banks, jewellers and casinos that deliver and receive goods of high monetary value. Due to the high value of the transported goods, the transportation process is exposed to large amounts of risk in terms of robberies or heists. The two main factors that are therefore considered in the RCVRP, is to minimise of the transport cost and to minimise the risk of exposure of the transported goods.

Even though high levels of security is not guaranteed to completely prevent robberies or heists, researchers speculate that the main reason why robberies on CIT vehicles are so common, is because of insufficient analysis on security issues during the route determination phase [121, 150]. In 2010, Ngueveu *et al.* [98] proposed that customers should be visited more often throughout a servicing window rather than delivering or collecting a large load on a single service run, however, vehicles are not allowed to traverse the same route more than once. This may be implemented by introducing time windows in the problem and, hence, introduce a certain level of *unpredictability*. Michallet *et al.* [91] have introduced a similar approach (also based on the VRPTW (2.44)–(2.53)) where the predictability of when customers are visited can be avoided.

In 2012, Yan *et al.* [150] formulated a cash transportation vehicle routing model which is based on the principles of an integer multiple-commodity network flow problem. This model is formulated by applying a time-space network technique in order to incorporate alternative transportation routes, while aiming to minimise operational costs. The model considers vehicle travel times, the number of customers that need to be serviced, as well as the number of vehicles that are available to service the customers. A number of assumptions are also included in the model and the assumptions were made after inspecting the possibility of implementing a RCVRP model in a real-world scenario that is implemented in the daily operations of Taiwan security transit company [150]. The first assumption is that the demand forecasts and customer locations may be known at least one day in advance, which ensures that customers and the time windows associated with customers are known and fixed. The second assumption made is that the fleet size is determined in advance based on the area which customers span and the number of customers considered in the problem. The third assumption is that each customer is only serviced by a single vehicle and the fourth assumption is that the scheduling horizon has a length of one day. Finally, it is assumed that the time that a vehicle spends at a customer and the fixed vehicle routes and schedules increase the risk of threats.

In practice, the typical elements considered in solving the transportation routing of high-valued goods and scheduling problems simply include the amount of valuable goods transported, the location of customers and their required service time [150]. According to Yan *et al.* [150], in real-world case studies, the most vulnerable moments of CIT vehicles are during the time that a vehicle is stationary at a customer or on a section of the route between customers. Therefore, in order to reduce the risk of a robbery or heist, an element of variation should be incorporated in daily routes and schedules allocated to these vehicles. The simplest form of incorporating variation into a model is to employ a *risk cost* for holding a vehicle at a customer. Yan *et al.* [149] modelled this as a constraint by proposing a similarity between time and space for all vehicle routes. Vehicles may therefore not service customers on the exact same time on the exact same day and the vehicles may also not travel the exact same routes at the exact same time each day. Let β_i denote the minimum allowable arrival time of a vehicle at customer i (in minutes) and let δ denote the maximum allowable similarity between space and time (in percentages). These two parameters are used to ensure variation in arrival times at customers. The vehicle arrival time at a customer is compared to the preceding days in order to ensure that the vehicle does not arrive at a customer at the same time that it did in the preceding days.

Suppose that a vehicle arrives at a customer A at `01:00pm` on a Friday. The arrival time of the vehicle is then denoted by β_{friday}. Therefore, the vehicle may not have arrived at customer A at

`01:00pm` during any of the five days preceding Friday (*i.e.* β_{sunday}, β_{monday}, β_{tuesday}, $\beta_{\text{wednesday}}$, β_{thursday}). The reason for this is that the arrival times are considered to be *similar* and, therefore unsafe. The parameter β_i is typically set as $\beta_1 \leq \beta_2 \leq \beta_3 \leq \beta_4 \leq \beta_5$, where $i = 1, \ldots, 5$ refer to the preceding day (*i.e.* where 1 denotes the previous day, 2 denotes two days prior, etc.) and, thus implying that the more recent the preceding day is, the larger the minimum permissible arrival time similarity must be. Furthermore, δ indicates whether the newly planned route and time schedule falls in the feasible range of δ or not (*i.e.* if the similarity falls within δ the plan is feasible, otherwise not). Suppose that the maximum allowable difference for Friday is $\delta = 0.6$. The maximum allowable difference for the preceding five days δ_1, δ_2, δ_3, δ_4, δ_5 may, therefore not be the same, since then they are regarded *similar* and considered unsafe. In general, the related δs are set as $\delta_1 \leq \delta_2 \leq \delta_3 \leq \delta_4 \leq \delta_5$, where $i = 1, \ldots, 5$ refer to the preceding day and, thus implies that the more recent the preceding day is, the smaller the maximum permissible similarity of time and space. It is, however, important to notice that as the number of customers on a single vehicle route increases, the number of similar routes that a single vehicle can possibly travel, also increases. This may result in the vehicle route similarity δ becoming more constrained over time. Yan *et al.* [149] explains how to calculate the similarity of space and time (β, δ) in more detail in his paper on "*A model with a solution algorithm for the cash transportation vehicle routing and scheduling problem*".

Based on the methods described above, Yan *et al.* [149] formulated a RCVRP model as an integer multiple-commodity network flow problem in which the objective is to minimise the operational cost of cash transportation in a network. Let $n \in \mathcal{N}$ denote a set of nodes where the node at 0 represents the depot, let $\mathcal{A}$ denote a set of arcs, and let $\mathcal{K}$ denote a fleet of vehicles. Let $t \in \mathcal{T}$ denote the days that are included in the problem for comparison for similarity, and let $\mathcal{P}_t$ denote the set of planned routes and schedules for day t. The set of arcs that form part of route k, where $K \subseteq \mathcal{A}$, that are similar to the arcs that form part of the planned route on day t may be mathematically expressed as $(K \subseteq \mathcal{A}) \sim \mathcal{P}_t$. Let this be denoted by $\mathcal{S}$ (note that $\mathcal{S}$ is derived from the concept according to which β is determined as discussed in the preceding paragraphs). Furthermore, let x_{ijk} denote the decision variable taking the value 1 if vehicle k travels from customer i to customer j, or the value 0 otherwise. The objective in the RCVRP of Yan *et al.* [149] is to

$$\text{minimise} \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{j \neq i, k=1}^{K} c_{ijk} x_{ijk}, \tag{2.55}$$

subject to

$$\left(\frac{\sum_{i=0}^{S} \sum_{j=0}^{S} x_{ijk}}{y_{pt}} \right) \leq \delta_{pt} \qquad i, j \in \mathcal{S},\ k \in \mathcal{K},\ t \in \mathcal{T},\ p \in \mathcal{P}_t, \tag{2.56}$$

where y_{pt} denotes the number of trips along the p^{th} planned route and schedule on the t^{th} day. Constraint sets (2.44)–(2.49) also form part of the RCVRP as in the VRPTW (2.44)–(2.53). Constraint set (2.56) ensures that each vehicle route and time schedule adheres to the constraint of maximum allowable similarity (δ) and that routes are, thus formulated in a less similar fashion.

In 2015, Talarico *et al.* [126] formulated a variant of the VRP which focuses on finding routes for vehicles in the CIT industry by introducing a risk-constraint. This problem is known as the *risk-constrained cash-in-transit vehicle routing problem* (RCTVRP). In the RCTVRP, a *risk threshold* is introduced to limit the risk of a robbery or heist on any route, where the risk is assumed to be proportional to the distance of the route travelled and the amount of valuable goods onboard the vehicle. In contradiction with the previous RCVRP formulations, Talarico *et*

al. [126] argue that in a practical sence (*i.e.* real-world scenarios) a natural variability in the exact moments does exist in time in which customers are visited. This is due to the variability in the amount of money that has to be delivered or collected at a given moment in time. In addition, most CIT companies calculate their vehicle routes on a daily basis according to the customers that require deliveries or collections on that day [126]. This results in some "unpredictability" since customer demands generally vary from day-to-day and, therefore, it is unlikely that the same set of customers will have to be serviced on the exact same days at the exact same times. Talarico *et al.* [126] therefore, formulated an approach that limits the accumulated risk that a vehicle may encounter on its route. Their formulation compliments the approaches in the literature suggested by other researchers and may be integrated with those approaches.

The RCTVRP also requires a fleet of vehicles that have to be optimally assigned to service a set of customers, each having a known demand — a common depot is also included in the problem. In RCTVRPs the demand is an amount of valuable goods have to be picked up from customers (no deliveries of valuable goods is included in the problem) [126]. Each vehicle has no valuable goods on board when it departs from the depot. The vehicle then collects cash from a set of customers on a route and returns to the depot after servicing (collecting valuable goods from) the customers. Since the amount of valuables in a vehicle increases as it services customers on a route each vehicle accumulates risk on its journey. The risk is proportional to the distance travelled and the amount of cash on board the vehicle. The total risk for each vehicle should therefore not exceed a predefined risk threshold. A predefined risk threshold is also typically incorporated in the problem which is typically determined by the CIT company by considering the amount of cash that will be collected on the routes, the characteristics of the given customer network (*how risky the surrounding area is*) and the company's attitude towards risk. There are three main attitudes towards risk [147]. The first is a risk *averse* attitude, where an organisation is not enthusiastic towards taking risk. The second is a risk *neutral* attitude, where an organisation is noncommittal towards taking risk and, finally, the third is a risk-*seeking* attitude, where the person is enthusiastic about taking risk. For simplicity and continuity, in the formulation of the RCTVRP, however, it is assumed that an organisation has a risk neutral attitude when making decisions.

The incorporation of risk in VRPs for the CIT industry has not received much attention, however, in the context of the transportation of *hazardous materials* it has been studied comprehensively. For the transportation of hazardous materials, a risk function is determined based on the characteristics of the substance that is being transported, as well as the characteristics of the route that is travelled [15, 140]. Furthermore, the *Center for Chemical Process Safety* [27] defines risk as an index by which economic loss, human harm and environmental catastrophe can be measured. The measurement may be based on the probability that a specific incident occursand the significance of the consequences of the loss, harm and catastrophe. Risk is, therefore classified as an *unwanted event* and may be expressed as

$$R_{\text{event}} = p_{\text{event}} \times C_{\text{event}}, \tag{2.57}$$

where p_{event} denotes the probability that an event will occur and C_{event} denotes the consequences of the event if it occurs. When an unwanted event occurs in the hazardous materials industry such as the spillage of chemicals in a public space, the consequences may be quite severe since the materials carried are typically very dangerous to the environment and people in the surrounding vicinity where the incident occurs. In contrast to the hazardous materials industry, the goods transported by vehicles in the CIT industry are not dangerous, but the occurrence of an incident such as a heist or robbery may trigger two types of events [126]. The first type is an event with *foreseeable consequences* for example in the event of a robbery occurring, the loss of cash is

expected. These type of events can be prevented by a CIT organisation by considering various security options such us using technologically advanced armoured vehicles with interlocking safes and vehicle tracking devices. The second type of event is an event with *unforeseeable consequences* for example is someone involved in the incident is killed, which are events that cannot necessarily be anticipated by the CIT company. The consequences of such events typically include damage to infrastructure or the loss human lives. These consequences of events typically depends on third parties, for example the agenda and skill level of robbers in a CIT heist. Since unforeseeable events cannot be predicted or analysed by a CIT organisation, they are typically not included in the formulation of RCTVRP. The RCTVRP formulations found in the literature therefore typically focus on minimising possible organisation financial loss of valuable goods transported [126].

According to Russo and Rindone [115] risk comprises three components. The first component is the *occurrence of an event*. Let p_{ij} denote the probability of an occurrence of a heist along the route between customer i and customer j, which provides an indication of how frequently an unwanted event occurs. The second component is the *vulnerability*, which provides an indication of the probability that a robbery will be successful. Let v_{ij} denote the probability that a robbery will be successful along the route between customer i and customer j. Finally, the last component is the *exposure*, which quantifies the loss of goods, harm towards people or damage towards infrastructure if the event occurs. Let D_i denote the probability of loss of goods if a heist occurs. Furthermore, Talarico *et al.* [126] assumed that a robbery may only occur once along a single route between customer i and customer j [126]. According to Talarico *et al.* [126] the risk that a vehicle can encounter on a single route may be calculated as

$$\sum_{i,j}^{N} p_{ij} \cdot v_{ij} \cdot D_i. \tag{2.58}$$

According to research conducted by Smith and Louis [121], criminals that typically target CIT vehicles are generally trained professionals who tend to be successful in their attempts to rob CIT vehicles and steal the valuable goods on board the vehicles. The assumption is therefore made that the vulnerability v_{ij} is constant for all attacks since the skill level of the criminals are assumed to be constant. Furthermore, it is assumed that criminals have no accomplice in CIT vehicles and, therefore, have no known knowledge of the amount of cash that is in the vehicle. It is also assumed that the probability of an attack p_{ij} is proportional to the length of the route c_{ij}. Talarico *et al.* [126] acknowledge that these assumptions might not always seem realistic, however, using the edge length as a measure of risk seems to be the most reasonable measure of risk since this information is always available. Moreover, no sufficient information or data exists in the literature that describes the characteristics of the routes in such a manner that it could be quantified realistically in order to anticipate risk. The route *risk index* may be formulated as a cumulative risk measure that is calculated by adding risk along the route that is travelled by a vehicle. Let R_i denote the risk index at the previous customer i and let c_{ij} denote the distance travelled between customer i and customer j and let D_i denote the demand collected at the previous customer i. The risk index R_j for the current customer may then be formulated as

$$R_j = R_i + D_i.c_{ij}, \tag{2.59}$$

From the risk index (2.59) it may be observed that the risk is proportional to the valuable goods collected from customer i (D_i) and the length of a route c_{ij}. The accumulated risk index for the

entire vehicle route may not exceed a predefined risk threshold. Let T denote the risk threshold defined by the user.

The objective in the RCTVRP is to minimise transport costs and the accumulated risk along a route and may be formulated as a mixed integer programming problem. Let x_{ijk} denote the decision variable taking the value 1 if route (i, j) is traversed by vehicle k, or the value 0 otherwise. The objective of the RCTVRP is therefore to

$$\text{minimise} \sum_{k}^{K} \sum_{i}^{N} \sum_{j}^{N} c_{ij} x_{ijk}, \tag{2.60}$$

subject to constraints

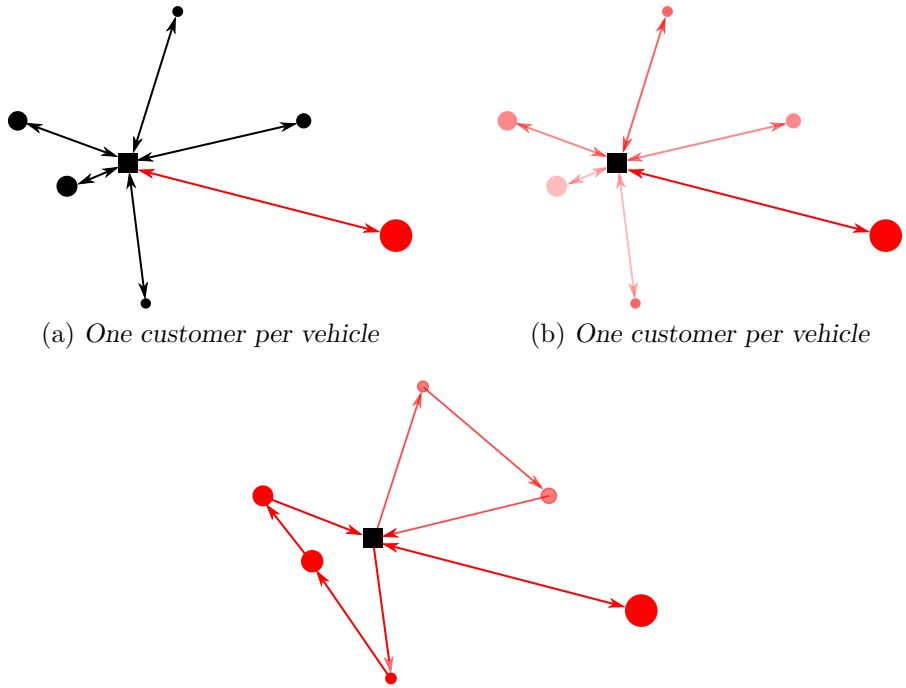
$$D_0 k = 0, \tag{2.61}$$
$$D_{jk} = D_{ik} + d_j - (1 - x_{ijk}), \qquad i, j \in \mathcal{V}, k \in \mathcal{K} \tag{2.62}$$
$$R_0 k = 0, \tag{2.63}$$
$$R_{jk} = R_{ik} + D_{ik} \cdot c_{ij} - (1 - x_{ijk}), \qquad i, j \in \mathcal{V}, k \in \mathcal{K} \tag{2.64}$$
$$0 \leq R_{ik} \leq T, \qquad i \in \mathcal{V}, k \in \mathcal{K}. \tag{2.65}$$

Constraint sets (2.27)–(2.32) from the CVRP model formulation also froms part of the RCTVRP model (2.60)–(2.65). Constraint sets (2.61)–(2.62) calculate the cumulative demand collected along the route, while constraint sets (2.63)–(2.65) calaculate the cumulative risk index for the route. The cumulative risk index is initially equal to zero when the vehicle departs from the depot at 0 and may not exceed the user-defined risk threshold T for any given route.

A number of benchmark problem instances of the traditional CVRP exist in so-called *VRP librabries* [63, 101]. These libraries contain problem instances of solved CVRP problems ranging in different levels of complexities and sizes and designs. These solved benchmark problem instances may then be used by other researchers to validate the models which they formulate and also to compare their results with the results form the benchmark problem instances.

There does, however, not exist a reference library with solved problem instances for the RCTVRP, due to its relatively new contribution towards the VRP. Talarico *et al.* [126] has, however, developed a library containing problem instances for the RCTVRP that is based on the original CVRP library. The difference is that Talarico *et al.* [126] added a risk threshold for each instance. They calculated a risk threshold for each of the problem instances by determining the minimum allowable risk threshold $T = max_{i \in N}\{d_i \times c_{i0}\}$ such that a feasible solution is found [126]. The minimum allowable risk threshold is determined by considering a case where a single vehicle is used to serve a single customer as illustrated in Figure 2.2(a). The route where a vehicle travels the furthest distance and collects the largest amount of valuables is the most risky route. In Figure 2.2(a) the size of the circle illustrates the amount of valuables that have to be collected at the customer (*i.e.* a larger circle resembles a larger demand) and the square denotes the depot. In Figure 2.2(a) it may be observed that the customer indicated in bright red is located the furthest form the depot and also has the largest amount of valuables that have to be collected, therefore, the bright red route will have the highest risk index. Figure 2.2(b) illustrates how the other routes have lower risk indexes by illustrating them in less intense variations of red. Thus, in order for the problem to have a valid feasible solution when including more than one customer on a single route — the lowest (or strictest) risk threshold that may be enforced on the problem must be more lenient than or equal to $max_{i \in N}\{d_i \times c_{i0}\}$. If the risk threshold was less than the

minimum allowable risk threshold, the vehicle would not be allowed to serve the customer in red (in Figure 2.2(a)) at all and the problem solution would therefore be infeasible. Figure2.2(c) illustrates how more customers may be added to the route as long as the accumulated risk index of the route does not exceed the risk threshold.



(a) *One customer per vehicle* (b) *One customer per vehicle*

(c) *Multiple customers per vehicle*

FIGURE 2.2: *The minimum allowable risk threshold that may be enforced on a problem.*

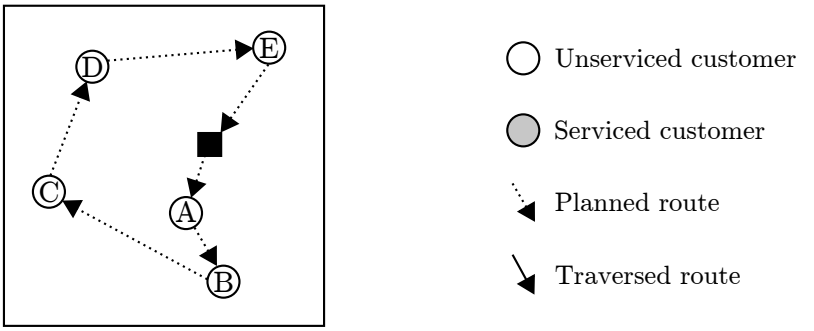
After the minimum allowable risk threshold is determined, each problem may be tested for sensitivity towards risk, respectively, by generating various risk thresholds that increase by a multiplicative factor in increments of 0.5, up to 3, [126]. An example is provided in Table 2.2, where the left column indicates the risk threshold name, and the right column indicates the risk threshold value. The minimum allowable risk threshold is determined as $T1 = max_{i \in N}\{d_i \times c_{i0}\}$. The minimum allowable risk threshold may then be relaxed (or made more lenient) by multiplying it with a factor 1.5 to achieve a risk threshold $T1.5 = 1.5 \times max_{i \in N}\{d_i \times c_{i0}\}$. Thus, $T1.5$ will allow more risk to be accumulated along a route than $T1$. Similarly, minimum allowable risk threshold may be relaxed by multiplying it with a factor 2 to achieve a risk threshold $T2 = 2 \times max_{i \in N}\{d_i \times c_{i0}\}$. Thus, $T2$ will allow more risk to be accumulated along a route than $T1$. The risk threshold relaxation may be iterated until the user is satisfied with the results.

Risk threshold	Value
T1	$max_{i \in N}\{d_i \times c_{i0}\}$
T1.5	$1.5 \times T1$
T2	$2.0 \times T1$
T2.5	$2.5 \times T1$
T3	$3.0 \times T1$

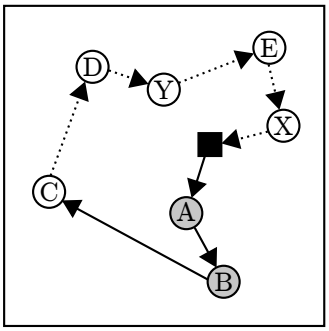
TABLE 2.2: *Calculation of risk threshold levels using the method of Talarico et al. [126]*

2.1.7 Dynamic VRP

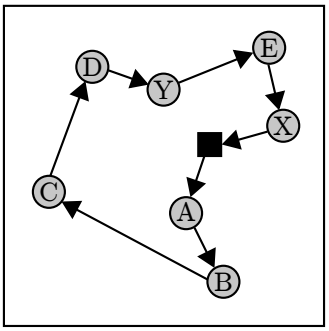
A recent development in VRP formulations is the formulation of the so-called *dynamic vehicle routing problems* (DVRP). Dynamic optimisation problems are problems that constantly change over the progression of time such as in the case of weapon assignment problems that continuously assess and fight threats, or a vehicle routing problem were new customers are continuously added to existing routes as the need arises. The formulation of the DVRP is mainly due to advances in information systems and communication systems that provide easy access to processed information in real time. Technological advances such as the introduction of global positioning systems (GPS) and geographical information systems (GIS), in conjunction with wide use of smart phones have resulted in an extensive multiplication of real-time routing applications [108]. In a typical DVRP, some orders are known in advance for a given day, but orders may also be placed throughout the day and have to be incorporated into the existing set of orders for the day. In the DVRP, the assumption is made that some type of communication platform exist between the customers and the supplier. The supplier is then responsible for periodically informing drivers of new customers that are assigned to their existing route(s).



(a) Vehicle route at time stage t_0.



(b) Vehicle route at time stage t_1.



(c) Vehicle route at time stage t_f.

FIGURE 2.3: *Example of the working of a DVRP for time instances t_0, t_1 and t_f, adapted from Pillac et al. [108].*

In order to illustrate the variety of the DVRP, consider the example in Figure 2.3. Suppose there are five customers (A, B, C, D , E) that have to be served by one vehicle, each denoted by a circle in Figure 2.3. The vehicle starts and ends its route at a common depot, denoted by a square in Figure 2.3. A white circle indicates that a customer has not been served by the vehicle, whereas a shaded circle indicates that a customer has been served by the vehicle. Furthermore, the dotted lines between customers indicates the suggested route for the vehicle, whereas the solid lines between customers indicates the actual route travelled by the vehicle. Figure 2.3(a) illustrates

the route for a single vehicle at time stage t_0 before it leaves the depot at time instance t_0. At time t_0, the route was planned to depart the depot and then travel to customers A, B, C, D and E. At the time instance t_1, two additional customers (represented by X and Y) required service from the vehicle, and these two customers were added to the existing route, as illustrated in Figure 2.3(b). The new route is therefore (after having serviced customer B) to serve customer C, D, Y, E and X. In this case, the supplier is responsible to communicate the change in the route to the delivery vehicle. Finally, at time instance t_f (illustrated in Figure 2.3(c)) the vehicle would have travelled from the depot to customers A, B, C, D, Y, E and X and returned back to the depot.

The dynamic VRP differs from static VRPs (*i.e.* the CVRP, the distance constrained VRP, the VRPTW, the PCVRP and the RCVRP discussed earlier) in numerous ways. According to Psaraftis [112] there are eight main differences. These differences are listed in Table 2.3.

Static VRP	Dynamic VRP
Time has no effect on the output result	The time dimension plays an integral role in the output result
Once the problem has started to execute, the output remains unchangeable	The output is adaptable once the problem has started execution
All inputs are known in advance	Unknown inputs are received and incorporated while the model executes
Does not require an information update mechanism during problem execution	Requires information update mechanisms that incorporates new information into the model as it is received
The route remains unchangeable once the problem is executed	Requires route re-sequencing or re-assignment procedures to incorporate new customers
Does not necessarily require faster computation times, because information is not necessary instantaneously	Requires faster computation times, in order to provide information instantly
Predefined vehicle fleet size	Flexible vehicle fleet size
No obvious need for queuing operations, except for the PCVRP where customers receive preference	Queuing operations could possibly be incorporated

TABLE 2.3: *Eight main differences between static and dynamic VRPs [112].*

Firstly, in a static VRP time has no effect on the output results achieved, while in a DVRP the time dimension plays an integral role in the output results that are achieved, because instantaneous change can be made to vehicle routes as the demand arises. In a static VRP, the output cannot be changed once the problem has started to execute, whereas in a DVRP, the output is adaptable due to instantaneous changes that occur during the problem execution. In a static VRP all inputs (*i.e.* variables and parameters) provided to the model are known in advance, whereas in a DVRP new inputs are received during the model execution. Due to the previously mentioned difference, the DVRP requires an information update mechanism that incorporates new information into the model execution in real-time, whereas the static VRP has no need for this mechanism. Furthermore, in the static VRP the routes remain unchanged once the problem is executed, whereas in a DVRP the routes may be re-sequenced after the problem is executed in order to incorporate new customers. Moreover, the DVRP requires fast computation times in order to effectively provide and incorporate new information instantaneously, whereas a static VRP does not require fast computation. The DVRP also allows for a flexible vehicle

fleet, whereas the static VRP has a predefined vehicle fleet and, finally, the static VRP has no obvious need for queuing operations (except for the PCVRP where certain customer are given selective preference), whereas the DVRP could possibly incorporate queuing operations.

In order to accommodate these differences, the dynamic VRP requires some design features such as an *interactive* platform to convey changes and new information, a calculation *restart* capability that is able to update routes instantaneously, a *hierarchical* design that enables the user to eliminate unnecessary problems before problem execution commences, and a *user-friendly* procedure that facilitates the interaction between man and machine.

In DVRPs, the waiting time of customers are often considered more important than the travel cost, for example the replenishment of stock in a manufacturing context, the management of taxi cabs, the dispatch of emergency services, and the replenishment of cash in ATMs. In most of these cases it is considered more important to service the customer as fast as possible rather than planning a cost efficient route.

Bertsimas and Van Ryzin [12] formulated a simple DVRP known as the *mobile repairman* problem. In this problem, a repairman is required to service a number of geographically dispersed failures. In the problem, it is assumed that the repairman is travelling at a uniform velocity in an area denoted by $\mathcal{A}$. Furthermore, all customer demands are dynamic in the sense that new demands can be added or removed instantaneously throughout the time period. Furthermore, the demand for each customer is given an arrival rate according to a Poisson distribution with an intensity parameter λ and the location of each failure is uniformly distributed across the area $\mathcal{A}$. Each failure requires a unique service time denoted by $\overline{s}$ and the variance of all the service times denoted by $\overline{s}^2$. Let ρ denote the fraction of time the repairman spends at a customer, which may be expressed as $\rho = \lambda\overline{s}$ and let T_i denote the time that has passed since the repairman has arrived at a customer until he has finished servicing a customer. In addition, let $T = \lim_{i\to\infty} E\big[T_i\big]$ denote the steady-state system time (*i.e.* the time a customer spends in the system, or the time it takes from when a customer enters the system until the time the same customer exits the system), and let W_i denote the waiting time from when the repairman has arrived at a customer until the repairman starts to service the customer. This implies that $T_i = W_i + s_i$ and $W = T - \overline{s}$. Bertsimas and Van Ryzin [12] emphasise that even though this system closely represents a queuing system, queuing theory may not be directly applied here since the DVRP system includes vehicle travel times which may not be viewed as independent variables. A number of routing approaches are discussed and documented by Bertsimas and Van Ryzin [12] and Larsen [77], and are as follows:

First come first serve (FCFS). In this approach, customer demands are serviced in the order that they arrive.

Stochastic queue median. This approach is a modification of the FCFS approach. In this approach the repairman operates form the median of the service area. The repairman travels directly to the demand location and thereafter back to the median location, while waiting for the next demand to arrive.

Nearest neighbour. After the repairman finishes service at a customer, he moves on to the closest next customer that requires service.

Travelling salesman problem. This approach batches customer demands in sets of n. Whenever a new set of customers is added to the problem, a TSP is used to solve the smaller problem involving the smaller subset n of customers. In the case where more than one set of customers exist, the sets are serviced according to the FCFS approach.

Space filling curve. Customer demands are serviced according to their location on a clockwise sweep of a circular region of the entire service area. Customers that appear on the sweep first, are serviced first, followed by the next customer that is encountered during a clockwise movement, until all the customers have been serviced.

Partitioning policy. The entire service area $\mathcal{A}$ is divided into sub-regions of square meterage. Each region is serviced individually by using the on the FCFS approach, and as soon as one sub-region has been serviced completely, the repairman moves on to the next sub-region.

Bertsimas and Van Ryzin [12] first establish lower bounds for the average system. Thereafter, they use a variety of techniques such as combinatorial optimisation, queuing theory, geometrical probability and simulation to analyse the policies discussed above and compare their respective performances to that of the lower bounds that were previously determined. This approach has proven that a variant of the FCFS policy (*i.e.* the *stochastic queue median* policy, where a vehicle waits at the median if there are no customers that require service) achieves optimal results in low traffic circumstances, while various methods achieve optimal results in high traffic circumstances. They consider two cases, namely a *light* traffic case and a *heavy* traffic case.

Light traffic: Light traffic implies that λ strives towards 0. In this case, the lower bound for T is established by considering three components of the system *i.e.* the waiting time incurred by travelling and servicing the customers prior to servicing customer i, the waiting time incurred from servicing customers preceding customer i and the individual service time of customer i. The best possible outcome is then

$$T^* \geq \frac{E\big[\ ||\ X - x^*\ ||\ \big]}{1-\rho} + \frac{\lambda \bar{s}^2}{2(1-\rho)} + \bar{s}, \tag{2.66}$$

where x^* is the median of $\mathcal{A}$. In the unique case where $\mathcal{A}$ is a square matrix, $E\big[\ ||\ X - x^*\ ||\ \big] = 0.383\sqrt{A}$ [77]. The bound T^* will therefore increase as the value of ρ increases.

Heavy traffic: Heavy traffic implies that ρ strives towards 1. In this case, Bertsimas and Van Ryzin [12] proved that there exists a constant $\gamma = \frac{2}{3}\sqrt{2\pi}$ so that the best possible outcome T^* may be estimated as

$$T^* \geq \gamma^2 \frac{\lambda A}{(1-\rho)^2} - \frac{1-2\rho}{2\lambda}. \tag{2.67}$$

This implies that the system time will grow by a factor $(1-\rho)^{-2}$, whereas if the problem was modelled as a typical queuing theory system time is expected to grow by a factor $(1-\rho)^{-1}$. The difference in these two approaches may be attributes to the existence of geometry in the service system.

Bertsimas and Van Ryzin [13] went further to generalise the findings in another paper for the capacited and uncapacited VRP cases. For the *uncapacitated* case with a fleet of k vehicles, travelling at a velocity v, there again exists two traffic cases *i.e.* light and heavy

Light traffic: The lower bound is expressed as

$$T^* \geq \frac{1}{v} E\big[min_{x_0 \in D*}\ ||\ X - x_0\ ||\ \big] + \bar{s}, \tag{2.68}$$

which provides the anticipated travel time form the closest repairman to a customer, including the on-site service time.

Heavy traffic: The lower bound is expressed as

$$T^* \geq \gamma^2 \frac{\lambda A}{k^2 v^2 (1-\rho)^2} - \frac{\overline{s}(1-2\rho)}{2\rho}, \tag{2.69}$$

which implies that the system will remain growing at a rate of $(1-\rho)^{-2}$. If, however, the number of repairman used or the velocity at which they travel is doubled, the lower bound is reduced by a factor 4.

In the case of the CVRP, each repairman is associated with a fixed unique depot location. Each repairman has an associated capacity q that may be used before the repairman is required to return to their depot. A lower bound for this case is provided by Bertsimas and Van Ryzin [13] as

$$T^* \geq \frac{\gamma^2}{9} \frac{\lambda A (1+\frac{1}{q})^2}{k^2 v^2 (1-\rho-\frac{2\lambda \overline{r}}{kqv})^2} - \frac{\overline{s}(1-2\rho)}{2\rho}, \tag{2.70}$$

where $\overline{r}$ denotes the anticipated distance from a customer to the nearest depot. In the case where the capacity q strives towards infinity, the bound is reduced to the expression (2.69).

Another important notion to consider in the DVRP is the *degree of dynamism* (dod), which is the ratio of dynamic requests received and the total requests received [83] *i.e.* it states how dynamic the problem is. The *dod* may therefore be calculated as

$$dod = \frac{\text{dynamic requests}}{\text{total requests}}. \tag{2.71}$$

The *dod*, however, does not consider the arrival times of dynamic requests. A planning system incorporating requests is, therefore, required. Consider a scenario where a planning system is constructed that starts at time 0 and ends at time T. Advanced requests are scheduled to be received prior to or at the beginning of the planning period (*i.e.* t_0). Let t_i denote instantaneous requests, where $0 \leq t_i \leq T$ and let n_{inst} denote the number of instantaneous requests received throughout the planning period. Furthermore, let n_{adv} denote the number of advanced requests that were received previously. The total number of requests may therefore be calculated as $n_{tot} = n_{adv} + n_{inst}$, and the *dod* measure may be redefined as the *effective dod* (edod) [77], and is calculated as

$$edod = \frac{\sum_{i=1}^{n_{inst}} (t_i/T)}{n_{tot}}. \tag{2.72}$$

The *edod* may be interpreted as the average time at which requests are received compared to the latest time requests that are allowed to be received. Note that

$$0 \leq edod \leq 1, \tag{2.73}$$

where 0 represents a dynamic system, and 1 represents a stochastic system. A long reaction time to a request is typically preferred by a planner of routes, and this may be incorporated by extending the *edod* to

$$edod - tw = \frac{1}{n_{tot}} \sum_{i=1}^{n_{tot}} \left(\frac{T - (l_i - T_i)}{T} \right), \tag{2.74}$$

$$= \frac{1}{n_{tot}} \sum_{i=1}^{n_{tot}} \left(1 - \frac{r_i}{T} \right), \tag{2.75}$$

where $edod - tw$ denotes the *edod* for problems with time windows l_i denotes the latest possible time service may start at customer i, and where r_i denotes the instantaneous reaction time of the vehicle to service customer i.

2.2 Solution approaches towards solving VRPs

The VRP has been studied extensively since the inception and many authors have contributed towards variants of the VRP since the initial formulation by Dantzig and Ramer [34] and the formulation provided by Clarke and Wright [29], through to the numerous heuristic approaches as surveyed by Laporte, Gendreau *et al.* and Cordeau *et al.* [31, 30, 76]. The reason why the VRP is so widely studied is because of it's versatility and efficiency in optimising operational costs in various distribution networks. There does not exist an exact algorithm that is capable of solving a VRP involving a large set of customers exactly within a reasonable computation time [96] and, therefore, using exact algorithms to solve VRPs are not considered practical for large problem instances [44]. This is because VRPs are known as *combinatorial optimisation problems*[3] and may be classified as *NP-hard*. In most cases, the most effective methods used to solve NP-hard VRPs include the use of heuristics.

In general, any Hamiltonian cycle that services each customer exactly once is considered as a feasible solution, however, the sought-after solution would be the Hamiltonian cycle with the lowest overall cost impact on the overall problem. Optimisation algorithms may roughly be divided into two main classifications of solution approaches that may be used to attempt to solve these problems, including *exact algorithms* and *heuristics*. Exact algorithms always find an optimal solution to an optimisation problem and typically include solution approaches such as *dynamic programming*, the *branch-and-bound* algorithm and the *branch-and-cut* algorithm. Exact algorithms are ideal for smaller problem instances, since they can be used to find exact solutions fast and with little effort. As the problem size grows, however, the time it takes to solve a problem using exact algorithms also increases exponentially. The drastic increase in computation time is commonly referred to as *combinatorial explosion* in the literature and refers to the problem that the number of combinations that require examination expand so fast that computers require an intolerable amount of time to examine all of the combinations [138]. Many researchers or computers, therefore turn towards the use of heuristic approaches to solve large problem instances and therefore exact algorithms will not be discussed further in this thesis. Most heuristics are able to achieve good solutions in a relatively fast computation time to achieve an immediate goal, however, not all solutions provided through the use of heuristics are necessarily optimal. Simple heuristics are commonly constructed with a specific problem in mind that perform a local search and only accepts improving solutions that may therefore be used to find a good solution for a specific problem. Simple heuristics terminate at the first good solution that is achieved. Typical examples of simple heuristic solution approaches include the *savings* method and the *sweep* method. These two methods are able to achieve good solutions that are acceptable, but not necessarily optimal. Meta heuristics, on the other hand, are problem-independent and may be used to solve a wide range of problems. Meta heuristics also tend to search further than local optima until some stopping criterion is met, however, they are still not guaranteed to find an optimal solution. Typical examples of meta heuristic solution approaches include the *simulated annealing* algorithm, the *tabu search algorithm* and the *genetic search* algorithm. The various methods available to solve VRPs are shown in Figure 2.4, where the

[3]Combinatorial optimisation problems aim to identify the maxima or minima of a given objective function with a discrete domain and broad configuration capacity [49].

methods indicated in red are the methods explored further in this section due to their popular nature.

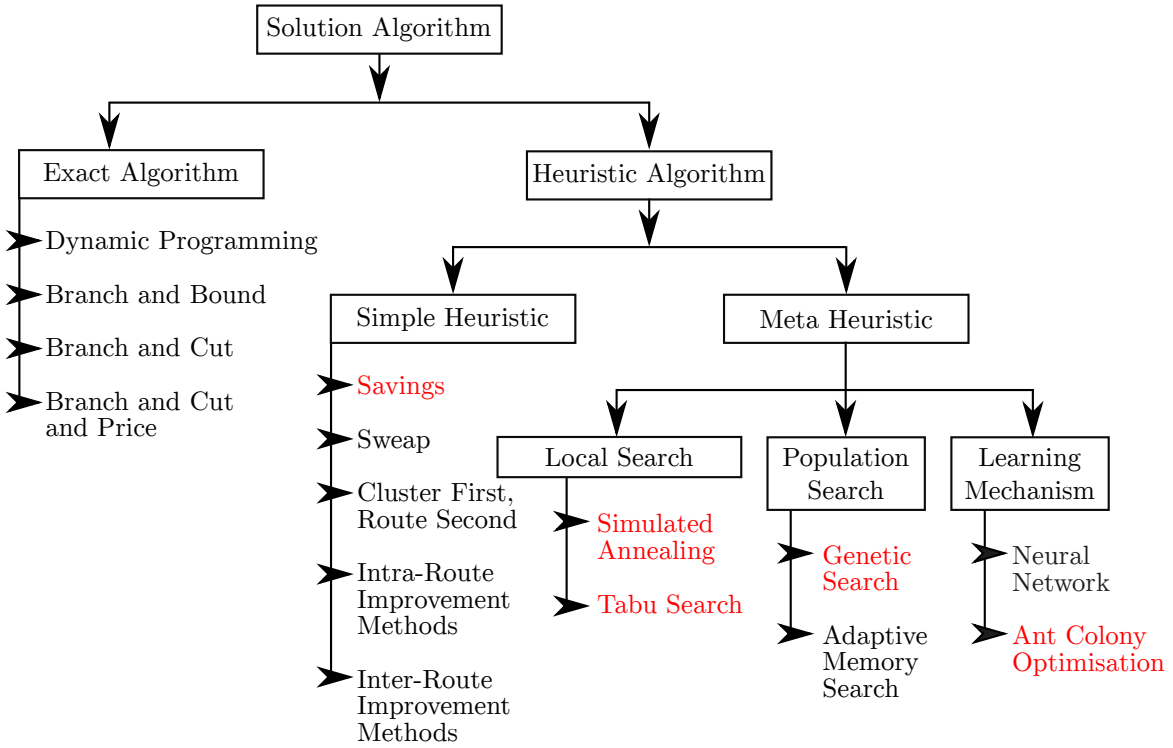


FIGURE 2.4: *Multiple solution methods that may be implemented to solve the VRP [96]*

2.2.1 Clarke-Wright savings heuristic

The *savings method* was first formulated by Clarke and Wright in 1964 [29] and is, therefore, commonly known as the *Clarke-Wright savings method* — which classifies as a heuristic solution approach. The method calculates a so-called possible saving for linking a specific pair of customers, when constructing a route. The method became the first algorithm that was widely used to find solutions for the classical VRP in (2.26)–(2.32) (as described in §2.1), after it achieved better results than the method proposed by Dantzig and Ramser [34]. This is due to its ability to find good quality solutions in a short period of time, as well as its simplicity and flexibility to accommodate a variety of constraints [44]. Dantzig and Ramser [34] introduced the first VRP formulation in 1959. Their algorithm formulation is based on a linear programming approach that considers linked combinations of customer pairs that are situated close together. Closely situated customers may be linked together and inserted into an existing route as a new customer pair if the addition does not violate any constraints. The customers are chosen based solely on the distance between them as a pair. Therefore, customer pairs that are located close together will be given preference and will be inserted into routes first. Clarke and Wright [29] extended on this method by considering the total reduction in distance of the *entire* route when inserting a linked pair of customers into a route, rather than serving the customers in the pair on separate routes.

The first step in the savings method is to calculate a so-called *saving* for each pair of customers in the network. The saving for each pair of customers is calculated according to the reduction in distance that may be achieved when linking two customers into a common route, rather than serving them on separate routes. A common depot is also included in the problem and is denoted

as node 0. Suppose that there exist two customers denoted by i and j that are located at a distance c_{i0} and c_{0j} from the common depot and located at a distance c_{ij} from each other. Two cases may be considered, as follows:

Case 1: Deliveries to each individual customer are made *separately*, while the distances travelled are assumed to be symmetric. This is illustrated in Figure 2.5(a), where a vehicle is required to travel the distance c_{0i} from the depot 0 to customer i and a travel distance c_{i0} travelling back to the depot. In addition, a vehicle is required to travel the distance c_{0j} from the depot 0 to customer j and a travel distance c_{j0} travelling back to the depot. The total distance TD_1 incurred may, therefore, be calculated as $TD_1 = 2c_{i0} + 2c_{0j}$.

Case 2: Deliveries to the various customers are made *simultaneously*, while the distances travelled are assumed to be symmetric. This is illustrated in Figure 2.5(b), where a vehicle is required to travel the distance c_{0i} from the depot 0 to customer i, a distance c_{ij} from customer i to customer j and then a distance c_{j0} from customer j back to the depot 0. The total distance TD_2 incurred may, therefore, be calculated as $TD_2 = c_{i0} + c_{0j} + c_{ij}$.

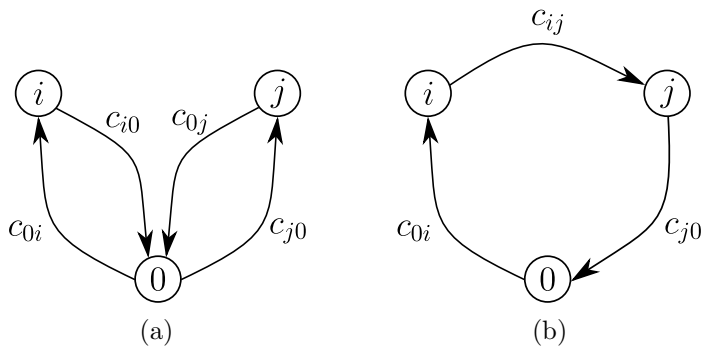


FIGURE 2.5: *Two cases for deliveries in the Clarke-Wright savings method. In (a) separate deliveries are considered and in (b) simultaneous deliveries are considered.*

By simultaneously serving both customers rather than servicing them on separate routes, a saving is incurred. Let s_{ij} denote the saving in distance that was achieved for linking customer pair i and j on to the same route [29]. The saving in the distance may then be calculated as

$$\begin{aligned} s_{ij} &= TD_1 - TD_2, && (2.76)\\ &= [2c_{i0} + 2c_{0j}] - [c_{i0} + c_{0j} + c_{ij}], && (2.77)\\ &= c_{i0} + c_{0j} - c_{ij}. && (2.78) \end{aligned}$$

In the Clarke-Wright savings method, equation (2.78) is used to calculate a saving s_{ij} for all the various pairs of customers i and j. These savings are then ranked in a list from best to worst by using the savings value (the saving achieving the highest value is at the top of the list and the saving achieving the lowest value is at the bottom of the list). A customer pair with a *higher savings* value is preferred when inserting customers into an existing route. A customer may only be inserted into a route if all the constraints (*i.e.* capacity and distance constraints) of the route are satisfied. If any of the constraints are violated, a new route is created and the customer pair is added to the new route until another constraint is violated. Furthermore, new customers may only be added to be serviced at the start or the end of existing routes — new customers are not allowed to be serviced in between existing customers.

The savings method may be implemented by using two different approaches, namely the *sequential* approach and the *parallel* approach. These two approaches are now discussed in more detail.

Sequential approach

In the sequential approach as described by Clarke and Wright [29], an arbitrary customer i is chosen as the initial customer in a route. Each time the algorithm is executed, the algorithm searches for a customer j that will have the next highest savings value when it is linked to the previously selected customer i. This is performed in an iterative manner until a constraint is violated. In the case where it is not possible to link another customer to the existing route, a new route is created until all customers have been inserted in a route. The steps for solving a VRP by using the Clarke-Wright savings method using a sequential approach are given as Algorithm 2.1.

Algorithm 2.1: The Clarke-Wright savings method using a sequential approach [29].

```
Initialisation;
for all customers do
    Create routes from depot to customer and back to depot in the form (0, i, 0);
for all routes do
    Calculate savings;
for all savings values in savings list do
    Sort savings in decreasing manner;
for all routes (0, i, ..., j, 0) do
    for each saving in savings list do
        if savings (s_ki) or savings (s_jl) may be added without breaking a constraint then
            Implement the merge in the form (0, k, i, ..., j, 0) or (0, i, ..., j, l, 0);
end;
```

Parallel approach

In the parallel approach, as described by Altinkemer and Gavish [4], routes are formed simultaneously by assigning an individual vehicle to a customer. Next, the savings for each of these routes are calculated and they are sorted in decreasing order of magnitude (*i.e.* from the highest savings value to the lowest savings value). The algorithm then forms new routes by merging existing routes from customers i and j that correspond to the highest savings value s_{ij}. This is executed in an iterative manner and in such a fashion that no constraints are violated or until no more routes can be merged. This approach therefore, allows route clusters to grow simultaneously. The steps for solving a VRP by using the Clarke-Wright savings method using a parallel approach are given as Algorithm 2.2.

Both of these approaches are able to achieve feasible results, however, they do not necessarily achieve the same results since the savings method is a heuristic that is able to produce feasible solutions which are not necessarily optimal solutions. Cordeau *et al.* [31] have, however, found that the parallel savings approach tends to produce better results than the sequential savings approach since the parallel approach solves the problem in an analogous manner rather than

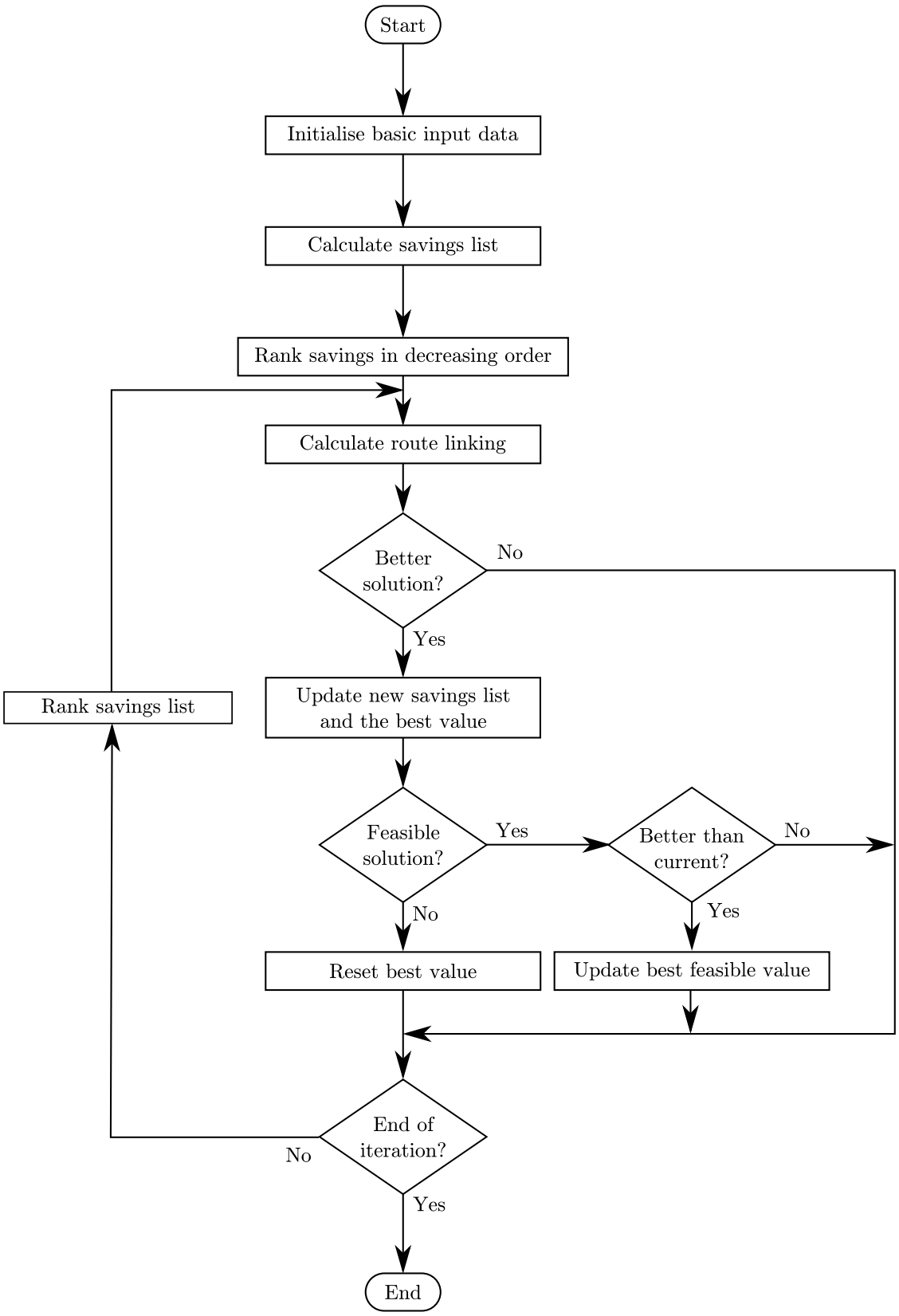


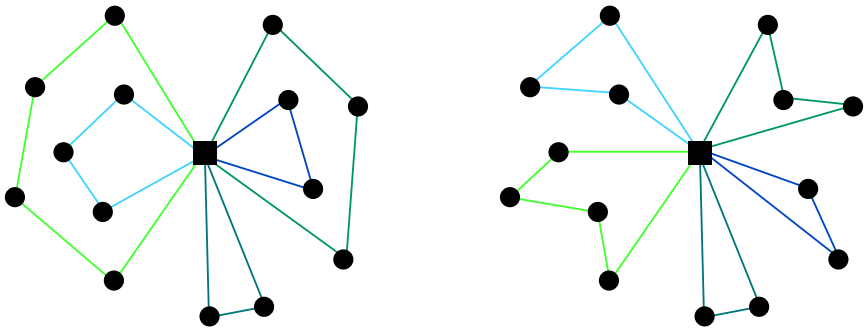
FIGURE 2.6: *A flowchart illustrating the steps involved in the Clarke-Wright savings method using a parallel approach, as adapted from Pichibul et al. [107].*

Algorithm 2.2: The Clarke-Wright savings method using a parallel approach [29].

```
1  Initialisation;
2  for all customers do
3  |  Create routes from depot to customer and back to depot in the form (0, i, 0) ;
4  for all routes do
5  |  Calculate savings;
6  for all savings values in savings list do
7  |  Sort savings in decreasing manner;
8  for each saving s_ij in the ranked savings list do
9  |  if there exists two routes (0, j, 0) and (0, i, 0) that can be linked without breaking
   |  constraints then
10 |  |  Link routes in the form (0, i, j, 0);
11 end;
```

sequentially. The steps in the parallel approach is therefore illustrated graphically in the form of a flowchart in Figure 2.6

Due to the nature of the Clarke-Wright savings algorithm, the algorithm is inclined to generate good quality routes during the early stages of executing the algorithm. This is due to the way in which the savings s_{ij} is calculated. Customers which are located further away from the depot will achieve higher savings values since they are located further from the depot. Customers located the furthest from the depot are therefore listed at the top of the savings list and are the first customers to be inserted into the route clusters. The algorithm tends to form circular-shaped routes that move from the furthest customers to the closest customers (with respect to the depot) as illustrated in Figure 2.7(a), rather than the familiar individual shaped routes that are typically associated with VRPs as illustrated in Figure 2.7(b).



(a) (b)

FIGURE 2.7: *The circular-shaped routes typically created by the Clarke-Wright savings algorithm in (a) and the typical individually shaped routes formed with other VRP solution approaches in (b).*

In 1967, Gaskell [53] and Yellow [151] claimed that the circular shaped routes formed by the Clarke-Wright savings algorithm may be seen as a disadvantage of the algorithm since it makes the routes unnecessarily longer. In order to reshape the route solutions to find better quality solutions, they introduced a parameter λ (having a positive value) to the savings formulation in the year 1970. The parameter λ was introduced in order to attempt to avoid the circular-shaped routes that are associated with Clarke-Wright VRP solutions (as illustrated in Figure 2.7(a)).

This changed the savings calculation in equation (2.78) to

$$s_{ij} = c_{i0} + c_{j0} - \lambda c_{ij}. \tag{2.79}$$

In 1988, Paessens [103] also modified the savings calculation (2.79) by introducing a second term with a parameter μ (also having a positive value) to the equation in (2.79). The inclusion of μ in the second term serves the purpose of exploiting the asymmetry of the distance between customers i and j and possibly improve the quality of the solutions. The savings calculation proposed by Paessens [103] is

$$s_{ij} = [\, c_{i0} + c_{j0} - \lambda c_{ij} \,] + [\, \mu \mid c_{i0} - c_{j0} \mid]. \tag{2.80}$$

Furthermore, in 2005, Altınel and Öncan [5] introduced a third term to the savings equation of Paessens [103] in (2.80) to include demand considerations of the customers. Their approach is based on a variation of the Knapsack problem (as described previously in §2.1.2). The savings equation proposed by Altınel and Öncan [5], therefore, gives preference to customers with larger demands and may be calculated as

$$s_{ij} = [\, c_{i0} + c_{j0} - \lambda c_{ij} \,] + [\, \mu \mid c_{i0} - c_{j0} \mid] + \left[\upsilon \frac{d_i + d_j}{\overline{d}} \right], \tag{2.81}$$

where d_i and d_j denotes the demands of customers i and j, respectively, and $\overline{d}$ denotes the average demand of all the customers in the network, while υ denotes a parameter (having a positive value) that is used to regulate the third term — the magnitude of the effect of the demand size on the equation is determined.

In 2011, Doyuran and Çatay [44] introduced a robust enhancement to the Clarke-Wright savings algorithm that is able to achieve improved computational results with respect to the distances travelled by each vehicle in the system. Their approach considers the way in which the savings s_{ij} is calculated and its working is based on the approach that was introduced by Altınel and Öncan [5]. The approach adopted by Doyuran and Çatay [44], however, contradicts the approach taken by Altınel and Öncan [5] in the sense that Doyuran and Çatay [44] argue that if smaller demands are given preference, the route solutions will also be improved when compared to the original approach by Clarke and Wright [29] where no preference is given to any specific demand. Doyuran and Çatay [44] modified the savings equation in (2.81) by subtracting the last term and in effect penalising customers with larger demands. The savings equation of Doyuran and Çatay [44] may therefore be expressed as

$$s_{ij} = [\, c_{i0} + c_{j0} - \lambda c_{ij} \,] + [\, \mu \mid c_{i0} - c_{j0} \mid] - \left[\upsilon \frac{d_i + d_j}{\overline{d}} \right]. \tag{2.82}$$

In addition, Doyuran and Çatay [44], proceeded to add a second modification to the formulation of Altınel and Öncan [5] in (2.81). Their variant of the savings equation gives preference to serve customers with a smaller demand first. They achieve this by using the inverse of the third term in the savings equation in (2.81). The second version of the savings equation of Doyuran and Çatay [44] may therefore be expressed as

$$s_{ij} = [\, c_{i0} + c_{j0} - \lambda c_{ij} \,] + [\, \mu \mid c_{i0} - c_{j0} \mid] + \left[\upsilon \frac{\overline{d}}{d_i + d_j} \right]. \tag{2.83}$$

Doyuran and Çatay [44] compared the savings equations that were proposed in (2.81)–(2.83) to test their working. They performed the test by comparing the three equations with one another, as well as benchmark VRPs from the literature. They also adjusted the parameter settings for λ, μ and υ in the intervals $[0.1,\ 2]$, $[0,\ 2]$ and $[0,\ 2]$, respectively. They concluded that the average deviation form the solution results do not have any significant difference when compared to each other [44], however, all three equations (2.81)–(2.83) outperform the original equation suggested by Clarke and Wright [29] and therefore, the equation suggested by the various researchers in (2.81)–(2.83) are notable considerations when employing the Clarke-Wright savings algorithm.

2.2.2 Simulated annealing heuristic

The method of simulated annealing was first proposed by Kirkpatrick *et al.* [71] in 1983. It mimics the annealing process in the strengthening of metals. The technique uses heat treatment to transform the molecular properties of a metal. The heat treatment process (also refered to as *annealing*) typically starts by heating a metal to a temperature above its recrystallisation temperature and, thereafter, the metal is allowed to cool down at a very slow pace. During the heating process, the atoms in the metal start to vibrate randomly and they advance through higher energy levels [14]. During the cooling process, the atoms in the metal start vibrating to a smaller degree until they resolve to a low energy state. Due to the slow nature of the cooling process, the atoms have a chance of resolving to an even lower energy state than the state they were in before the annealing process started.

Simulated annealing algorithms, therefore, imitate this process by accepting *non-improving* moves during local searches according to some probability in order to possibly achieve a new and better solution. The algorithm starts by considering an initial solution and then iteratively examining and performing some functions on a single solution, before moving on to the next. This method is referred to as a *trajectory-based* method. Thus, simulated annealing is a method for solving combinatorial optimisation problems. In the case where the goal is to minimise the cost function, the *optimal* solution would be the solution that minimises the cost function the most within a global context. Therefore, when any solution is provided, the algorithm will attempt to improve the given solution by making incremental local changes (*i.e.* explore the neighbouring solutions), this local optimisation is visually illustrated in Figure 2.8.

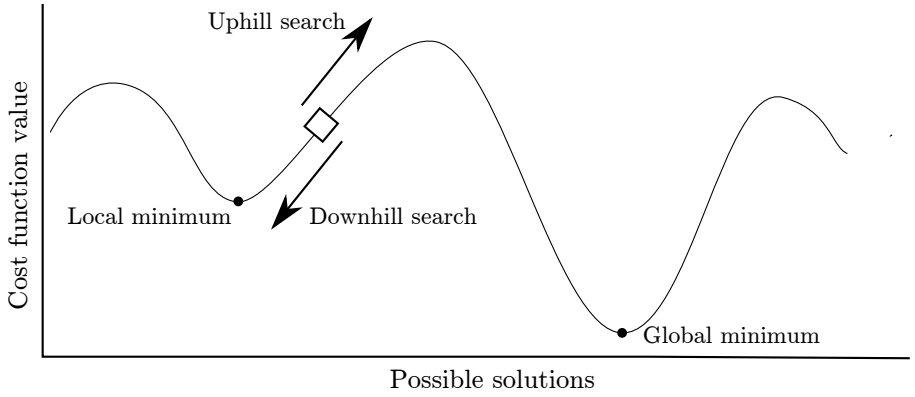


FIGURE 2.8: *An illustration of the variation of the cost value function when optimisation is applied in a local environment.*

Simulated annealing attempts to incorporate an occasional uphill move (*i.e.* a non-improving move) in order to attempt to overcome a local optima solution. This is incorporated by using a parameter referred to as the *temperature*, which is influenced by a random number generator.

The process of the simulated annealing algorithm is illustrated in Figure 2.9. The algorithm requires an initial solution as input in (1), as well as a *temperature limit* denoted by T_{end} and an initial temperature denoted by T_{begin}. In (2) the algorithm checks that the initial temperature $T > 0$ and if the initial temperature T_{begin} is larger than or equal T_{end}, the algorithm terminates. If the initial temperature T_{begin} is less than T_{end} in (3), the algorithm continues to step (4). If the iteration limit is reached in (4) the algorithm terminates, however, if the iteration limit has not been reached the algorithm continues to consider neighbouring solutions. In (5) the algorithm chooses a random neighbouring solution according to a problem-specific set of possible moves (this specific problem type typically has a particular combinatorial context under consideration). The change in temperature is calculated between the new neighbouring solution and the previous good solution and is denoted as Δ in (6). If the move results in an improved objective value (*i.e.* $\Delta > 0$) in (7), the move will always be accepted as in (8a). If, however, the move does not improve the objective value (*i.e.* $\Delta \leq 0$), the move may be accepted according to some probability known as the *Metropolis rule* [139], *i.e.*

$$e^{-\Delta C/T_i}, \tag{2.84}$$

where ΔC denotes the change in the objective value when a move is made from the current solution to the new solution and where T_i denotes the temperature value at epoch i of the search. An epoch denotes a series of successive iterations where the temperature is kept constant. If the move is accepted according to the probability, it becomes the new current solution for the next iteration as in (8b), however, if the move is rejected, the algorithm explores the next neighbouring solution according to the problem-specific set of possible moves. The process form (3)–(8) is repeated until acceptance occurs. Acceptance occurs if T_i is less than T_{end} or if the iteration limit is reached.

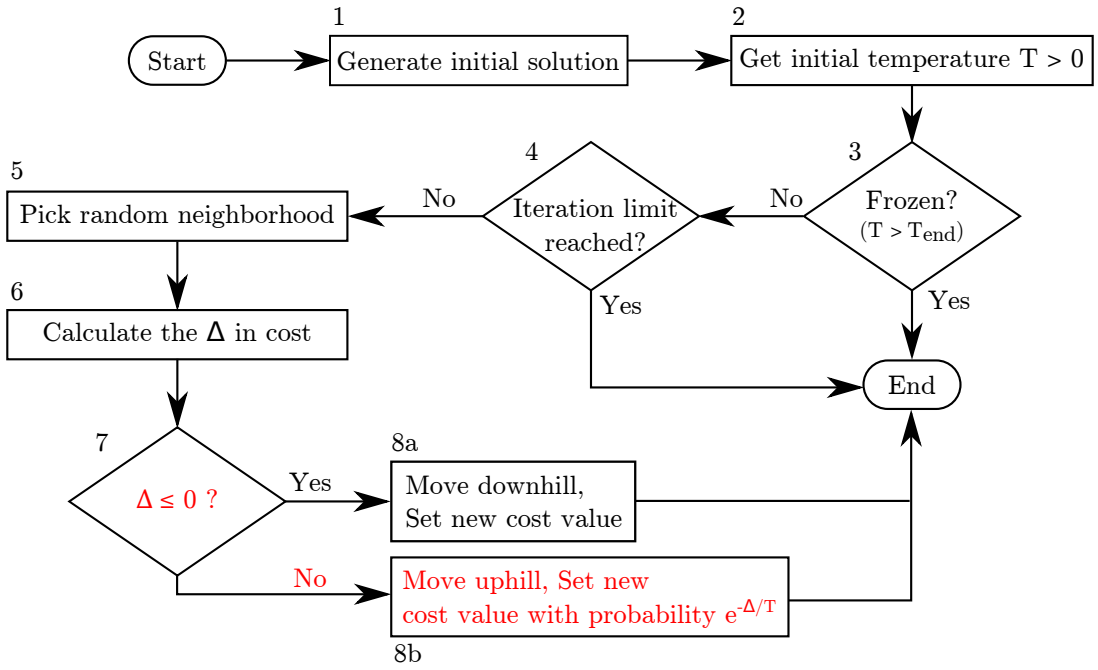


FIGURE 2.9: *A flowchart illustrating the working of the simulated annealing method.*

Busetti [23] suggests that the initial temperature should be selected in such a manner that at least eighty percent of the non-improving moves are accepted at the start of the search. The initial temperature T_{begin} may be approximated by conducting an experiment search where all of the non-improving moves are accepted. Then, the average change observed between all of the

objective values that come forth as a result of this experiment may be determined and denoted as Δ^+. Then, a good starting value for the initial temperature T_{begin} may be approximated by calculating

$$T_{begin} = \frac{-\Delta^+}{\ln 0.8}. \tag{2.85}$$

Every epoch has a maximum number of iterations associated with it and, therefore, the length of all epochs will not necessarily be the same. The number of iterations the temperature remains the same is typically determined by using a Markov chain with length denoted as L_i and should typically be specific to the problem that is solved [23]. Let $A_{\min}$ denote the minimum number of move acceptances that are allowed before the temperature may be lowered and before the next epoch may commence. Non-improving moves are accepted with a decreasing probability as the iterations $i \xrightarrow{\infty}$ the temperature $T_i \xrightarrow{0}$ and, thus, the number of experiments necessary before accepting the value of $A_{\min}$ becomes unbounded (*i.e.* very large) as the search continues. Thus, let L denote the number of experiments after which an epoch is terminated or when $A_{\min}$ is accepted, while $L > A_{\min}$. Dreo *et al.* [45] suggest that L may be equal to 100 and that $A_{\min}$ may be equal to $12N$, where N denotes the number of degrees of freedom that is associated with the specific problem at hand.

There are various cooling schedules that may be employed (for example linear, geometric and adaptive schedules), however, the geometric schedule is the most popular [142]. A geometric function is proposed for the cooling function, where the temperature at each iteration may be denoted as $t \in \{1, \ldots, T_t\}$. The cooling schedule for the current iteration may therefore be calculated as

$$T_t = \alpha T_{t-1}, \tag{2.86}$$

where α denotes the cooling rate (ranging between 0 and 1). Furthermore, reheating schedules may be employed in a similar fashion as the cooling schedule, however in the case if the heating schedule the temperature for each epoch is increased by some factor denoted by β (larger than 1). Reheating is aimed at making it easier to accept worsening solutions in order to promote the possible escape from a local optima. The reheating schedule for the current iteration may therefore be calculated as

$$T_t = \beta T_{t-1}. \tag{2.87}$$

In the context of vehicle routing, simulated annealing has been proposed by various researchers in different varieties. Osman [102] proposed a hybrid of the simulated annealing and tabu search metaheuristics, which was used and adapted by Thangiah *et al.* [135] in order to address VRPTWs combined with pick-ups and deliveries. Furthermore, Teodorovic and Pavkovic [133] used simulated annealing to generate initial solutions, as well as to solve VRPs with stochastic demands.

2.2.3 Tabu search heuristic

The tabu search heuristic is considered as a generalisation of iterative improvements that makes use of adaptive memory capabilities and strategic decision making — it is classified as a meta heuristic. This allows the algorithm to use a combination of various methods, such as linear

programming and specialised heuristics that prevent the algorithm from becoming trapped in local optima. A neighbourhood is identified in the search within which adjacent (neighbouring) solutions may be explored and considered as possible candidate solutions to the problem. Tabu search implements restrictions that guide the search over a diverse search space by using adaptive memory and responsive exploration. To illustrate this concept, consider for example, an individual climbing a mountain. The individual may remember attributes of the path being travelled (adaptive memory) and thereby make strategic decisions on the return route (responsive exploration). When making certain decisions, the outcome is, therefore, interpreted and the next decision is made according to the outcome of the previous decisions. In such a case, it is considered beneficial to make bad strategic decisions, since a bad strategic decision might reveal valuable information than an average good strategic decision. The adaptive memory of the tabu search algorithm may be explained according to four dimensions, namely: quality, recency, frequency and influence [109]. *Quality* of the memory refers to the ability to differentiate between good and bad characteristics of a decision in order to ultimately be able to penalise a bad decision. *Recency* refers to the short-term memory that keeps track of previous solution attributes and how they have had an impact on the change in outcome during the recent past. *Frequency* and *influence* ensure that a certain measure of diversity exists when deciding on solutions in the long-term in order to guide the solution to explore a part of the unvisited solution space that most likely may contain promising solutions. A primary way of exploiting memory would be to classify a subset of the solution space moves as *forbidden* (*i.e.* also reffered to as *tabu*) and, therefore, these moves are added to a so-called *tabu list*. Tabu moves are, however, subject to a noteworthy exception — in the case where a tabu move presents a possible appealing evaluation such that it could possibly result in a better solution overall, then the "tabu" classification may be overridden. This is then typically referred to as an *aspiration move* since it aspires to achieve a better objective function value. Furthermore, the current solution is refered to as the *seed*.

One of the first applications of the tabu search to the VRP was done by Willard [146], with the goal of providing higher quality solutions that are "close-to-optimum" within a reasonable computational time. Since its inception, tabu search has been researched and refined by many researchers to become the preferred solution approach when solving problems involving artificial intelligence and optimisation. A flowchart containing the basic working of the tabu search algorithm is illustrated in Figure 2.10. The steps are as follows in the method [59]

1. *Initialisation.* The algorithm requires an initial solution which is set as the initial seed as well as the current best solution (since it is currently the only solution).

2. *Stop criterion satisfied?* The current best solution is evaluated and if the stopping criterion is met, Step 3 is executed. If the stopping criterion is not met, the algorithm moves on to Step 4. Typical stopping criteria that may be used are

   - that there exists no feasible solution in the neighbourhood of the current solution,
   - that the maximum number of total iterations have been reached,
   - that the maximum number of consecutive iterations without improvement have been reached, and
   - that there is an indication that an optimal solution has been reached.

3. *Output solution.* If Step 3 is reached, the algorithm terminates and the solution found thus far is returned as solution output to the problem instance.

4. *Generate neighbours.* The algorithm is used to consider neighbours of the current solution by exploring the search space. This is done by applying a local transformation to the current solution, for example in the context of a CVRP problem, a single customer that forms

part of a route in the current solution, may be moved and inserted into a different position in the same route, or into another route altogether — given that the new transformed solution remains valid. Typical insertion methods include

- random insertion, which randomly inserts the selected customer into a random position in the current route or in another route, or
- insertion at the best position, which inserts the selected customer into a the best position the target route.

5. *Aspiration criteria satisfied?* The new neighbour solution is evaluated to check whether it is valid and better (*i.e.* not tabu) than the current best solution. The algorithm uses both a *forbidding* and *freeing* strategy depending on whether a solution should be added to the tabu list. If the aspiration solution is valid and better than the current best solution, the algorithm moves on to Step 6, otherwise if the solution is not improving on the current solution, the algorithm moves on to Step 7.

6. *Update.* If the aspiration criteria in Step 5 is met, a new "best" solution has been found and this is then updated throughout the algorithm and, therefore, it is *freed* (removed) from the tabu list.

7. *Select best neighbour.* If the aspiration criteria in Step 5 is not met, the considered aspiration criteria was invalid (*i.e.* tabu) and the algorithm, therefore, considers the next best valid solution from the neighbourhood which is not tabu, and then it is selected to become the new seed.

8. *Update tabu list.* The tabu list prevents the algorithm from visiting previously considered solutions (*i.e.* it serves as the algorithm's short-term memory) and, therefore contains solutions that are *forbidden*. These solutions may typically not be revisited within a predefined number of iterations. If there are too little entries in the tabu list, the algorithm may cycle through the same solutions, however if there are too many entries in the tabu list it creates too many restrictions and, therefore, limits the possible moves of the algorithm. Setting the ideal length of the tabu list remains an open problem in the literature, and is typically determined through trial and error [153]. An experiment suggests that it is best to allow the tabu list length to vary dynamically throughout the search [153].

2.2.4 Genetic search heuristic

The *genetic algorithm* (GA) was first invented by Holland [61] in 1992. It is based on Darwin's theory of evolution and mimics the notion of natural selection. In the GA, a population of individuals (or chromosomes) are maintained by allowing them to evolve over time by using natural selection techniques and mutation operators. The chromosomes are allowed to evolve over a number of iterations (that represent time) with the aim of discovering good solutions to an optimisation problem. During each iteration, parent solutions are chosen from the given population based on their respective fitness values. The fitness measure used is based on a pre-defined criterion and the fitness value provides an indication on the quality of the solution. Parent solutions are typically selected from the population by a *selection* operator. Once parent solutions have been identified, a *recombination* operator is applied to produce offspring solutions. The offspring solutions then, typically replace their parent solutions as the next generation of possible solutions. The selection and recombination techniques are applied iteratively throughout the different generations in order to attempt to find the fittest possible solutions. Once significantly

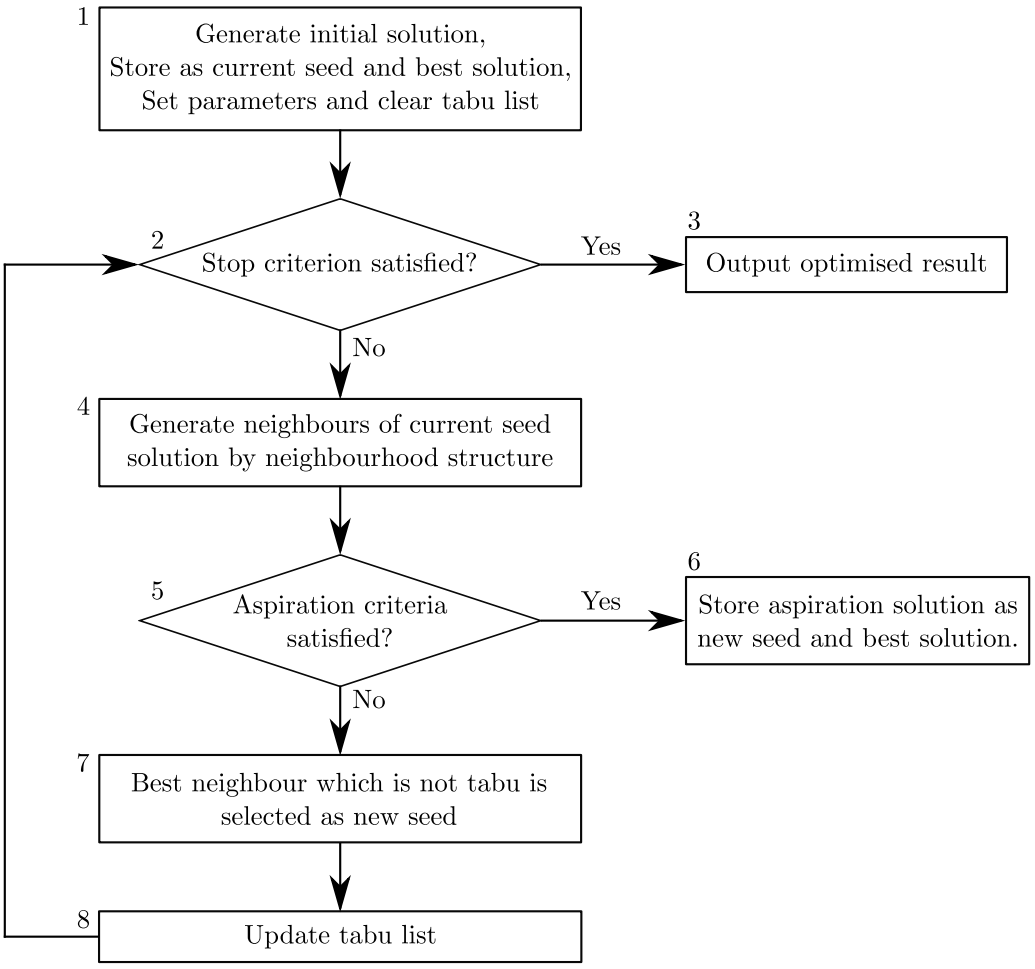


FIGURE 2.10: *A generic flowchart of the tabu search algorithm. Image adapted by Zhang et al. [153].*

fitter solutions cannot be identified the algorithm terminates. Figure 2.11 illustrates an example that the components namely populations, chromosomes and genes used in a GA.

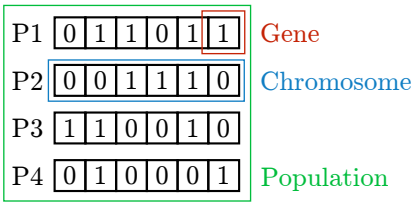


FIGURE 2.11: *The three components used in the GA.*

The working of the algorithm is illustrated using a flow chart in Figure 2.14. In (1) the algorithm requires inputs such as the size of the population denoted as N, the maximum number of allowed iterations that the algorithm may be executed denoted as $t_{\max}$, two recombination probabilities denoted as p_c and p_m respectively, and the size of a route S_t. Furthermore, the algorithm requires an initial possible solution population denoted as $\mathbf{P}^0$ which has a size equal to N. In (2) the fitness of all possible solutions in the population are determined. If the iteration limit (*i.e.* generation limit) has been reached in (3), the algorithm terminates. If, however, the iteration limit (*i.e.* generation limit) has not been reached in (3), the algorithm continues by choosing two parent solutions from the population in (4). According to Dreo *et al.* [45], a popular method used to choose two parents from the population is the *tournament* method. The method consists of a

random small subset of solutions from the population, which represents a *tour*. The number of solutions is the small subset is refered to as the *tour size*. From the tour, one solution is selected to be included in a so-called *mating pool* as one of the parent solutions. The total number of parent solutions present in the mating pool constitute the *pool size*. The chosen solutions are typically selected based on their respective fitness value — solutions with higher fitness values are more likely to be chosen. The selection procedure may typically be executed according to a method known as the *roulette wheel selection* method. The procedure calculates the probability of a chromosome's selection based on its fitness. Let f_i, therefore, denote the fitness value of chromes i, then the probability of its selection may be calculated as

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i} \tag{2.88}$$

The procedure, therefore, normalises the fitness value of the chromosomes in the population. Then, a circular-shaped wheel is divided into sections which are divided by arcs, where every arch represents an individual chromosome. The fitness value of the chromosome is used to determine the span of each arc (*i.e.* like a roulette wheel in a casino — a chromosome with a higher fitness value will have a wider arc and, therefore, a higher probability of being chosen). Furthermore, a fixed point is established on the wheel and it is then rotated. The chromosome on the wheel that corresponds to the pre-defined fixed point is selected as the first parent for crossover. The selected parent is removed from the wheel (also the population) and then the wheel is rotated again to select a second parent for crossover.

Once the two parent solutions are identified they undergo a crossover procedure in (5) in order to produce two new offspring (*i.e.* children) in (6). The crossover procedure in (5) is linked with the probability p_c, where $p_c \in (0, 1)$. The probability of crossover is typically chosen to be large, because crossover is an integral method used to introduce some variation into the population of solutions [73]. Figure 2.12 illustrates the notion of crossover. The left chromosomes illustrate the parent solutions, while the right chromosomes illustrate the offspring generated through the crossover procedure. It may be observed that the both offspring solutions constitute different combinations of the two parent solutions.

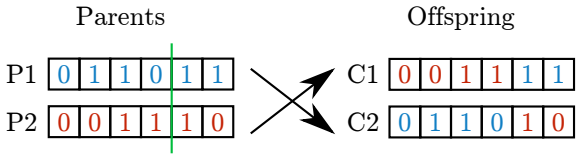
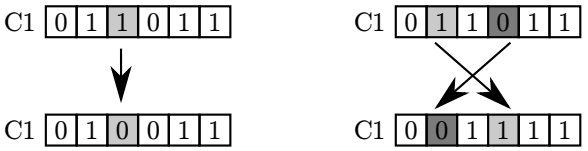


FIGURE 2.12: *The crossover process of a GA.*

After the offspring are generated they may be diversified even further by performing a mutation procedure which is linked to a probability p_m. The aim of the mutation procedure is to avoid local convergence to a solution that is sub-optimal — it, therefore, promotes an occasional further search. The probability of mutation (p_m) is typically selected to be small in order to avoid the search becoming random [73]. If the probability of mutation is too low in (7), mutation does not take place and the algorithm moves on to (9). If the probability of mutation is high enough in (7), mutation will take place in (8). Figure 2.13 illustrates the notion of mutation. The mutation method typically involves the process of inverting a binary bit (*i.e.* a gene) as is illustrated in Figure 2.13(a) where the gene with the binary value 1 in $C1$ is inverted to a gene with a binary value of 0. Another method that may be used to preform mutation is by interchanging two genes in the same chromosome as is illustrated in Figure 2.13(b) where the

second gene in the chromosome $C1$ is interchanged with the fourth gene in the same chromosome in order to produce a new solution.



(a) *Inverting mutation.*

(b) *Interchanging mutation.*

FIGURE 2.13: *An inverting mutation in (a) and an interchanging mutation in (b) for use in the genetic algorithm.*

In the context of solving VRPs using a GA, after crossover and mutation have occurred in (5) and (8), each offspring is evaluated by using a so-called *reparation* process in order to check whether each chromosome (*i.e.* route) contains the correct number of genes (*i.e.* customers on the route) in (9). If there exists a case where a route services a single customer more than once, or a case where a customer is not included in the route at all, another process has to be executed to correct (or repair) this. The algorithm *removes duplicate* customers from the route and also *inserts missing customers* into the route according to some *penalty value*. This penalty value in (10) is used to rate the fitness of a chromosome and, therefore facilitates distinction between good routes and bad routes. The hardness of the penalty is decided upon before the algorithm is executed and, in order to decide whether an offspring is accepted or rejected, its penalty value is compared to that of its parents. If the offspring solution achieves a better penalty than it's parents in (11), it is accepted as part of the next generation (*keep child*). However, if a child performs worse than it's parents in (11), it is subject to a second selection process. The purpose of the second selection process is to provide another opportunity for preserving some diversity and, therefore champions the parents according to their penalty values and provides the offspring solution with a chance of survival according to some probability (*keep parent*).

Next, the fitness of each child in the new population generation is calculated to determine whether they should be allowed to repopulate. This is achieved by calculating the fitness, expressed as

$$\text{fitness} = 100 \times \frac{\text{Maximum penalty} - \text{Chromosome penalty}}{\text{Maximum penalty}}. \tag{2.89}$$

The fitness value serves as a local measure of fitness for the current generation under consideration. This value is used to select participants for the next generation where candidates with higher fitness levels are more likely to be selected.

The GA is iterated in this fashion until a maximum number of stopping criterion is reached. Typical stepping criteria that may be used include the maximum number of iterations is reached, a predefined fitness value, or until a satisfactory solution is reached [88].

2.2.5 Ant colony optimisation heuristic

Dorigo *et al.* [43] was the first to described the *ant colony metaheuristic* in 1992 after observing ants and their behaviour when searching for food. Their goal was to solve combinatorial optimisation problems based on this behaviour of ants. While observing how ants behave daily,

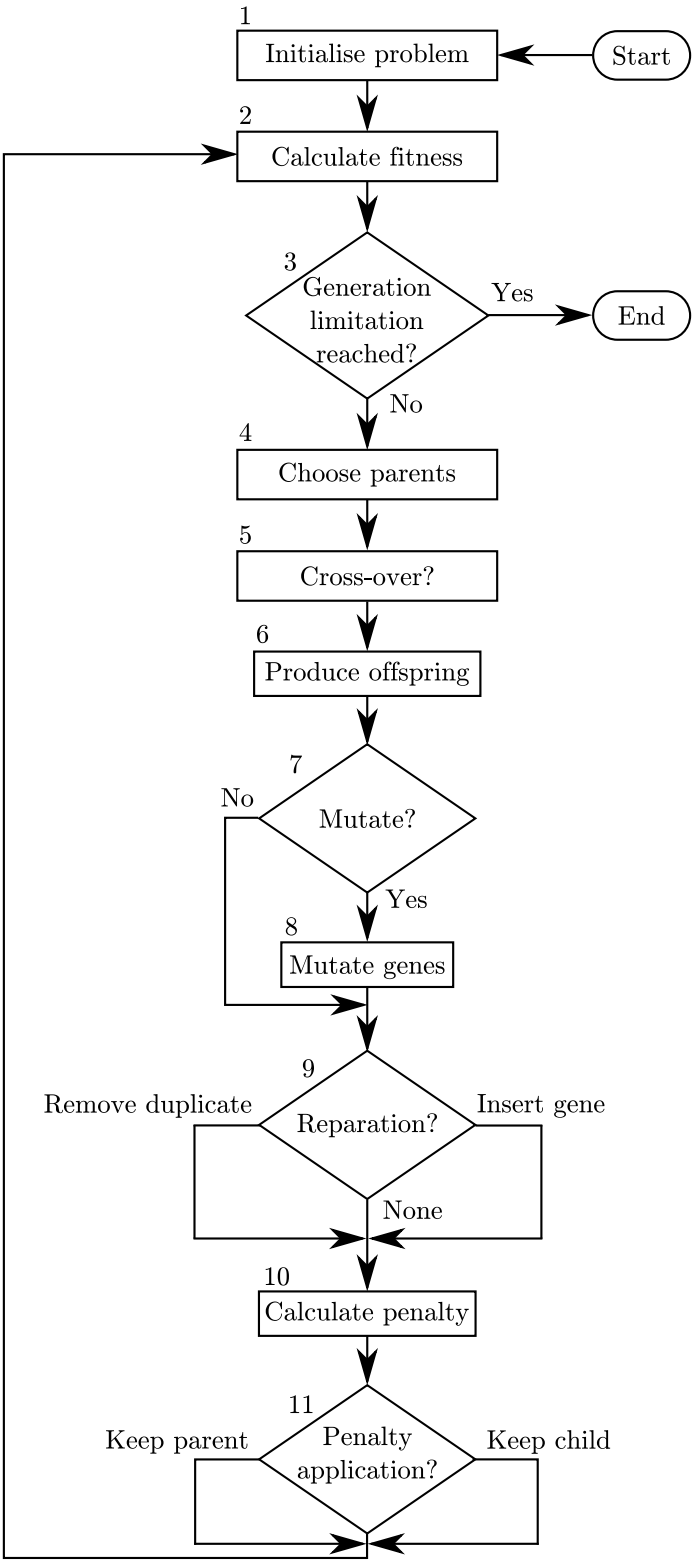


FIGURE 2.14: *A flowchart of the process involved in a GA, as adapted from Masum et al. [88].*

it was found that they have the ability to communicate with one another with respect to the locations of food, by releasing an aromatic fragrance known as *pheromone*. Ants release this pheromone while travelling to a food source and other ants are able to pick up the scent which may decide to follow the trail to the food source. This phenomenon is called *stigmergy* and

is defined as the interaction between individual organisms by modifying their environment and thereby creating complex structures and behaviour without making use of planning or any form of direct communication [58]. Therefore, ants that are roaming lonely and randomly are able to pick up on these pheromones and join other ants on a trail to a food source. When an ant joins a trail, it reinforces the pheromone on the trail once again, resulting in an even stronger pheromone. A trail becomes more attractive as more ants follow the trail since the pheromone becomes stronger on the trail and, therefore, the process can be represented as a positive feedback loop where the probability of an ant choosing a given route becomes higher as the number of ants on the route increases.

Deneubourg *et al.* [37] researched the release of pheromones and behaviour of ants in a renowned experiment known as the *double bridge experiment*. During this experiment, a colony of ants were connected to a food source with two equal length bridges, and then bridges with varying lengths were enforced. Initially ants randomly chose a path to follow to the food source, as was anticipated. After allowing the ants to collect food from the source for some time, however, it was observed that all ants have converged to traversing the shortest path. This phenomena can be explained by understanding that the short bridge has a quicker travel time compared to bridges with longer path lengths. Therefore, ants that travel along the shortest bridge reach their destination quicker and deposit pheromone on the trail more regularly than ants travelling on longer routes, thus making the shorter route more preferable. This exploration technique adopted by ants allows them to transport food in an exceptionally fast and effective manner.

The double bridge experiment served as a starting point for the development of the first simple stochastic model for the ant colony algorithm. The behaviour of the model was developed in 1990 by Deneubourg *et al.* [37] by assuming that ants cross a bridge at a constant speed and deposit one unit of pheromone on the bridge during their crossing. Furthermore, it was assumed that the movement of an ant is dependant on a probabilistic choice of path, where this probability is a function of the pheromone present on the given path (*i.e.* proportional to the number of ants which have chosen the path). Let m_1 denote the number of ants that traversed the first path and let m_2 denote the number of ants that traversed the second path. The probability p_1 that ants will choose the first path may then be calculated as

$$p_1 = \frac{(m_1 + k)^k}{(m_1 + k)^h + (m_2 + k)^h} \ , \tag{2.90}$$

and

$$p_2 = 1 - p_1, \tag{2.91}$$

respectively, where k and h represent parameters that are fitted according to experimental data. In the literature presented by Dorigo *et al.* [42], *Monte Carlo simulation*[4] was used to estimate $k \approx 20$ and $h \approx 2$. These parameters may be fluctuated in order to vary the impact of the pheromone on the shortest path, as well as the congestion present in the model.

In 1992, Bullnheimer *et al.* [22] used the ant colony algorithm to solve VRPs (as described in §2.1) by considering a VRP with a single depot and a homogeneous fleet of vehicles. Bullnheimer's approach to solving the VRP [22] is based on the formulation used by Dorigo for the TSP [43]. Artificial ants are used in the model to represent the behaviour of the real ants, bearing in mind

[4]Monte Carlo simulation is a mathematical technique used to model random input variables according to some probability distribution that is then used to model risk or uncertainty in a system.

that a few primary differences exist between artificial ants and real ants according to Blum [17]. These differences include that

1. Real ants move in an asynchronous fashion in their environment, whereas artificial ants move in a synchronous fashion within their simulated environment (*i.e.* in the simulated environment, ants move from their nest to the food source and back to their nestby following the same path).

2. Real ants leave pheromone behind whenever they move anywhere, whereas artificial ants will only deposit pheromone on their way back to the nest (*i.e.* once they have found the food source and can move with certainty).

3. The foraging behaviour of real ants is based on the constant evaluation of a solution (*i.e.* a solution is a path from the nest to the food source. The constant evaluation refers to the fact that the shortest path will be completed quicker than longer routes and, therefore, the shorter path's pheromone intensity will increase more rapidly). On the other hand, artificial ants evaluate a situation by considering a quality measure that indicates the strength of the pheromone reinforcement that an ant will provide on it's path back to the nest.

The basic working of the ant colony algorithm is described in the form a flow chart in Figure 2.15. First the algorithm parameters have to be initialised in step 1. Assume that the VRP of Bullnheimer is structured similar to the graph G described in §2.1, with the depot located at node v_0 and the remaining nodes representing customers that require service. The distance (or cost or time) between customer i and customer j is denoted by c_{ij}. Each customer i has a non-negative demand denoted by d_i and service time denoted by δ_i, while assuming that $d_0 = 0$ and $\delta_0 = 0$ for the depot. The aim in the ant colony algorithm in this context is to minimise the overall cost when allocating vehicles to routes that service a network of customers. Each customer may only be visited exactly once, all vehicle routes are required to start and end at the common depot, each vehicle has a capacity limit of Q, and each vehicle route may not exceed a route length of L. Furthermore, an evaporation coefficient denoted as ρ is initially defined for the ant colony algorithm . The evaporation coefficient is very important during the initial stages of the search, as it prevents convergence to a weak solution (or local optima), however, the coefficient should allow the pheromone to fade sufficiently and become almost negligible in the later stages of the iterations in order to be able to provide a clear solution as the end of the search approaches. For the initial placement of ants in the system, it has been found that the initial number of ants in the system should be equal to the number of customers that have to be serviced, and that one ant is initially placed at each individual customer at the start of the iteration.

Following the initialisation of the basic ant colony algorithm, there are two steps that are repeated for a specified number of iterations, namely the *creation of vehicle routes* (2) and the *trail update* (3). During the second step in the algorithm, the ants are randomly allocated to every customer in the system and are required to move to the next customer according to a probability distribution that is employed according to the amount of pheromone present on a route. In order to solve the VRP, artificial ants denoted by $k \in K$, where $k = 1, \ldots, K$ represent the vehicles used to choose which customers to service until all customers have been serviced. In the case where the next chosen customer returns an infeasible solution (*i.e.* vehicle capacity Q is reached or route limit L is exceeded), the vehicle returns to the depot and an additional route is introduced. For a vehicle to decide which customer to visit next, two aspects have to be considered. The first aspect is whether the choice is considered as a *good* choice according to the

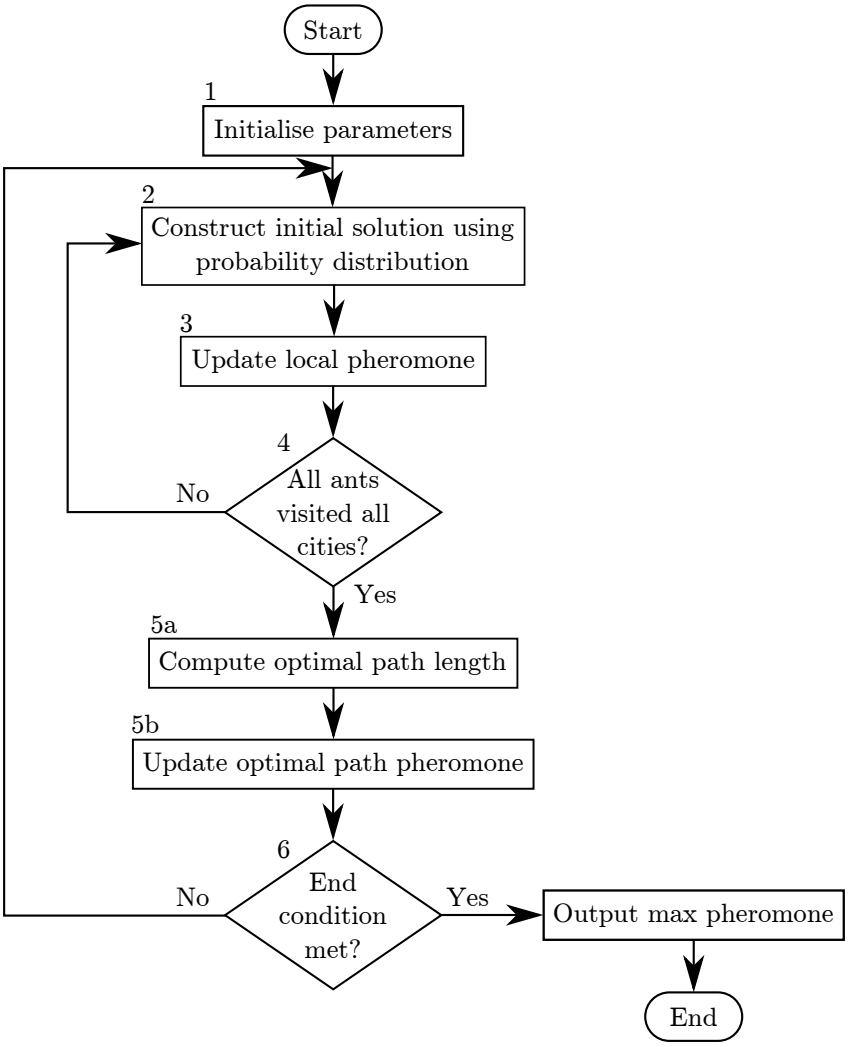


FIGURE 2.15: *Flowchart of the ant colony optimisation algorithm, as adapted from Yun et al. [152].*

strength of the pheromone trail. Let τ_{ij} denote the strength of the pheromone trail associated with each route between customer i and customer j. The second aspect is whether the choice is considered *desirable*. Let η_{ij} denote the desirability of a route between customer i and customer j, where $\eta_{ij} = 1/c_{ij}$. Each customer is selected based on a transition probability function given as

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{j=1}^{m}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}} & \text{if } v_j \in \Omega \\ 0 & \text{otherwise,} \end{cases} \tag{2.92}$$

where Ω denotes all the customers that are feasible to visit next. The parameters α and β are derived from τ_{ij} and η_{ij}, respectively, and provides a trade-off option to the user on the importance of each aspect. In the case where $\alpha = 0$, the probability p_{ij} is solely based on heuristics, which means that only cities that are close to each other will be chosen and, therefore, the algorithm will act as a greedy algorithm. In the case where $\beta = 0$, the probability p_{ij} is solely based on the pheromone concentration of a route which is not considered ideal, since the algorithm can become trapped within a localised search space. Pheromone values are not allowed to be equal to zero since division by zero is prohibited in transition probability function (2.92).

The pheromone trail is laid and determined according to the objective function value of each tour L_k that was created by a previous passing ant. The pheromone trail is updated or increased by $\Delta\tau_{ij}^k$ for each route (v_i, v_j) that was traversed by any given ant k.

Furthermore, let L^* denote all routes that currently belong to the best known solution. These routes are emphasised and is said to be travelled by *elite ants*. One elite ant is able to increase the pheromone intensity on a trail by $\Delta\tau_{ij}^*$, thus making that route even more favourable than an average artificial ant would have.

The third step in the flowchart of Figure 2.15 is that the pheromone intensity levels on the trail have to be updated with each iteration by updating the function

$$\tau_{ij}^{new} = \rho\tau_{ij}^{old} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \tag{2.93}$$

where ρ is the evaporation coefficient and m is the total number of ants in the system. Thus, the initial number of ants in the system should be equal to the number of customers that have to be serviced ($m = N$).

If all ants in the system have not visited all the customers at this stage, the algorithm iterates steps 2 and 3 again until all ants in the system have visited all the customers after which it may continue to step 5. During step 5 the current path length for the solution is calculated in 5(a) and the pheromone intensity of the route is updated in 5(b). If the end conditions are met in step 6, such as the final number of iterations or a satisfactory solution, the algorithm may provide the maximum pheromone route intensity (*i.e.* the best route solution) as an output and terminate, otherwise if none of the end conditions are met, the algorithm returns to step 2.

The ant colony algorithm has been proven effective to solve combinatorial optimisation problems, however, Ding *et al.* [41] explained that there are some disadvantages associated with using the approach. The first disadvantage is that the search easily becomes trapped in local optimum. The second disadvantage is that a substantial amount of computational time is required to find an optimal solution and, finally, the parameters are not easily adjustable to obtain best performance.

2.3 Chapter summary

This chapter was dedicated to VRPs from the operations research literature. The history and evolution of the VRP since its inception in 1959 was discussed and summarised in §2.1. The first, and most basic formulation of the VRP is the TSP and was discussed in §2.1.1. Many variations of the VRP model have been formulated over the years and typically include additional constraints such as capacity, distance, time, risk and dynamic constraints. These variations of the VRP model were discussed individually in §2.1.2–§2.1.7. The VRP is a combinatorial optimisation problem and, therefore, the focus of the chapter shifted to the solution approaches that may be used to solve the VRP. Theses solution approaches include the use of heuristics or meta heuristics. Some of the more popular solution approaches that may be used to solve the VRP were named in §2.2 and include the working of the Clarke-Wright savings algorithm, the simulated annealing algorithm, the tabu search algorithm, the genetic algorithm, and the ant colony algorithm, respectively. Each of these solving approaches were discussed in sections §2.2.1–§2.2.5, respectively. The chapter finally closed with a brief summary on the chapter contents.

CHAPTER 3

Simulation literature study

Contents

This chapter serves as an introduction towards simulation modelling and contains various aspects of the modelling technique from the operations research literature. The chapter opens in §3.1 with an introduction towards simulation modelling. A definition and some basic history on simulation in general is provided. Hereafter, in §3.2 the various paradigms of simulation modelling are discussed, with a focus on static versus dynamic simulation models, deterministic versus stochastic simulation models and continuous versus discrete simulation models. Next, in in §3.3, the levels of abstraction (including high, medium and low abstraction) according to which simulation models may be developed are mentioned. Simulation modelling methods such as discrete event modelling, agent-based modelling, system dynamics modelling and dynamic systems modelling are also described in §3.3. In §3.4 various elements employed to replicate a real-world problem are discussed. Next, in §3.5 the advantages and disadvantages associated with simulation modelling are discussed. The twelve steps involved in developing a simulation model is mentioned in §3.6 and, lastly, there is a discussion on the difference between optimisation and simulation in §3.7. The chapter finally closes in §3.8 with a brief summary of the chapter contents.

3.1 Introduction to simulation modelling

Simulation concepts were first formulated in the 1770s by a Frenchman named Buffon, who is famous for his *needle problem*. The needle problem is defined as the probability that a needle of length l may be dropped on a floor space that is decorated with equally spaced, parallel lines (that are located a distance d from each other) and intersect one of the lines. This problem serves as the first example of an experiment that requires independent replications of a simulation experiment in order to approximate a physical constant [54]. Later, in the 1940s, a Polish mathematician, named Stanisaw Ulam, had access to one of the first electronic computers and determined that it may be used to effectively estimate the complex mathematical integrals associated with the design of a hydrogen bomb [54]. In the 1980s, Ulam collaborated with John von Neumann and Nicholas Metropolis to develop the now famous *Monte Carlo simulation method* that is widely used to approximate random input variables [54]. Simulation has been researched by many researchers over the past three centuries and has been found to be applicable in a wide range of application areas to date.

The art of simulation modelling is, therefore, a very popular and widely used operations research optimisation technique that is employed in many industries. Simulation may be defined as the imitative representation of an existing (or proposed) system's behaviour or characteristics through the use of a computer program. Simulation, therefore, enables a user to predict the future behaviour of a system, by taking into account its current state. This is particularly beneficial when it is necessary to understand how a certain system will act under changing circumstances over an extensive period of time.

A simulation model may be experimented with and reconfigured according to a decision maker's criteria, whereas in practice these types of changes typically have considerably large costs associated with them if they are implemented sub-optimally or completely incorrect. Before constructing a simulation model, it is important to fully understand the problem that is simulated, as well as what should be expected in the outcome of the study. The characteristics concerned with the behaviour of a real-world system and the subsystems contained therein may be derived from studying the working of the model in a simulated environment. Subsequently, it may be used in a variety of other applications, such as reducing the likelihood of failing to satisfy system specifications, eliminating unanticipated bottlenecks, or preventing over or under utilisation of allocated resources, as well as optimising the overall performance of the system [86].

Simulation modelling typically requires a set of assumptions and objectives in the form of mathematical or logical relationships that are constructed to portray and examine the behaviour of a system [78]. In the case where these assumptions and relationships are uncomplicated, it might be possible to solve the problem mathematically in order to determine an *exact* solution, however, in most real-world problem instances the problems are too complex to solve exactly (for example manufacturing systems, transportation systems and business re-engineering). In these cases, simulation is considered an invaluable solution methodology used to find a well *estimated* solution to the problem at hand.

In this section, a number of existing types of simulation approaches from the literature, as well as the context in which each of these model types may typically be applied are discussed. Simulation paradigms are described in terms of their respective abstraction levels (*i.e.* the level of detail and strategic intent). The various elements and role-players required to build a successful simulation model are also listed and defined.

3.2 Simulation paradigms

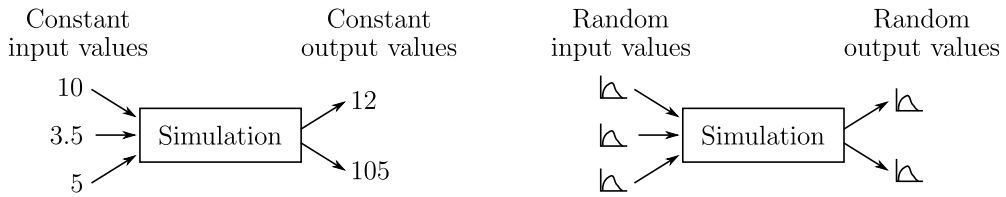
Simulation models may be categorised into three different paradigms, depending on the level of detail and strategic intent it incorporates. These paradigms are considered as *static versus dynamic* simulation models, *deterministic versus stochastic* simulation models and *continuous versus discrete* simulation models [78].

3.2.1 Static versus dynamic simulation models

A *static* simulation model (*i.e.* an analytic simulation model in the sense that it relies on a mathematical approach) may be described as a system that is inspected only at particular time-steps in a single run of the simulation, or a system where time is an irrelevant characteristic and the focus rather lies on statistical outcomes such as in the case of the celebrated Monte Carlo simulation [57]. This type of model is typically dependant on user-specified input parameters that are used to provide sensible statistical output. In contrast, a *dynamic* simulation model requires a progression over time in order to change state variables in an implemented function over time. An example of a dynamic system is a manufacturing production line where the demand of the manufactured product changes continuously over time.

3.2.2 Deterministic versus stochastic simulation models

A *deterministic* model does not have random input variables and, therefore, all future states in the model are resolved as soon as the initial inputs and relationships have been specified. Solving such a type of model may, however, be very time consuming [57]. On the other hand, a *stochastic* model takes random inputs and produces corresponding random outputs, which as a result hold some degree of uncertainty. Therefore due to the random nature of this model, it is mainly used to find estimated solutions [78]. A typical example of a stochastic simulation model is a system involving arrival times or flow rates [57]. The constant values of the input and output of a deterministic simulation model is illustrated graphically in Figure 3.1(a), while the random input and output values of the stochastic simulation model are illustrated graphically in Figure 3.1(b).



(a) *Deterministic system* (b) *Stochastic system*

FIGURE 3.1: *The deterministic input and output values of a deterministic simulation model in (a) and the random input and output values of a stochastic model in (b). Adopted from Harrell [57].*

3.2.3 Continuous versus discrete simulation models

Discrete event simulation is a simpler, less detailed solution approach and may be applied to a number of applications [86] such as a customer arriving at a bank teller and requesting

services, or a truck arriving at a loading dock expecting to load goods. Changes to discrete-change state variables are triggered by events at a certain point in time for example, failures and arrivals in the context of a production line. Discrete events are documented by making use of discrete-change variables that are used to indicate when an event takes place. This is illustrated graphically in Figure 3.2. The state of a model, therefore, becomes the cumulative state of the states of all the components of the model at a discrete point in time. In contrast, the state variables of a *continuous* model change continuously as time progresses. An example of a continuous system is the level of water present in a tank at any given time instance, while water is loaded or unloaded from the tank. These systems typically make use of differential equations or inequalities to specify rates of change in the system as a function of time [57]. The state variables of a continuous simulation model are illustrated graphically in Figure 3.2.

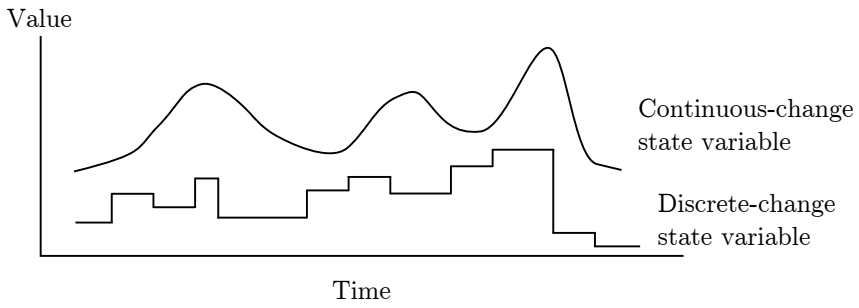


FIGURE 3.2: *The state variables of a discrete-event system in comparison with that of a continuous-event system. Adopted from Harrell [57].*

3.3 Levels of abstraction of simulation problems

Each simulation problem has different approaches that may be followed and depends on the difficulty of the problem being solved — each problem holds a certain level of abstraction. These levels of abstraction are illustrated in Figure 3.3. High abstraction involves problems with less details (*i.e.* problems are typically based on aggregates, trends or influences), and is presented on a macro scale with strategic tendencies. Medium abstraction involves problems with medium details, presented on a mesco scale with tactical tendencies. Finally, low level abstraction involves problems with more detail (*i.e.* problems are typically based on actual values, exact sizes, real-time and specific individual objects), and is presented on a micro scale with operational tendencies.

The four main paradigms in simulation modelling are *discrete event* modelling, *agent-based* modelling, *system dynamics* modelling and *dynamic systems* modelling [21]. Agent-based modelling is quite a new development in the simulation realm (it only became widely used in the late 1990s), while system dynamics modelling and dynamic systems modelling have been around since the 1950s. The four paradigms are discussed in further detail in the remainder of this section.

Discrete event modelling models a system as it progresses over time, where events occur instantaneously at various discrete points over the time period and has a medium level of abstraction [78]. These events are occurrences that may ultimately influence the state of the system. This type of simulation requires less detail than continuous event simulation, which makes it easier to implement and, therefore it is more widely used. Some of the characteristics associated with discrete event simulation, as listed by Rouse [114], include a predefined start and end time (as

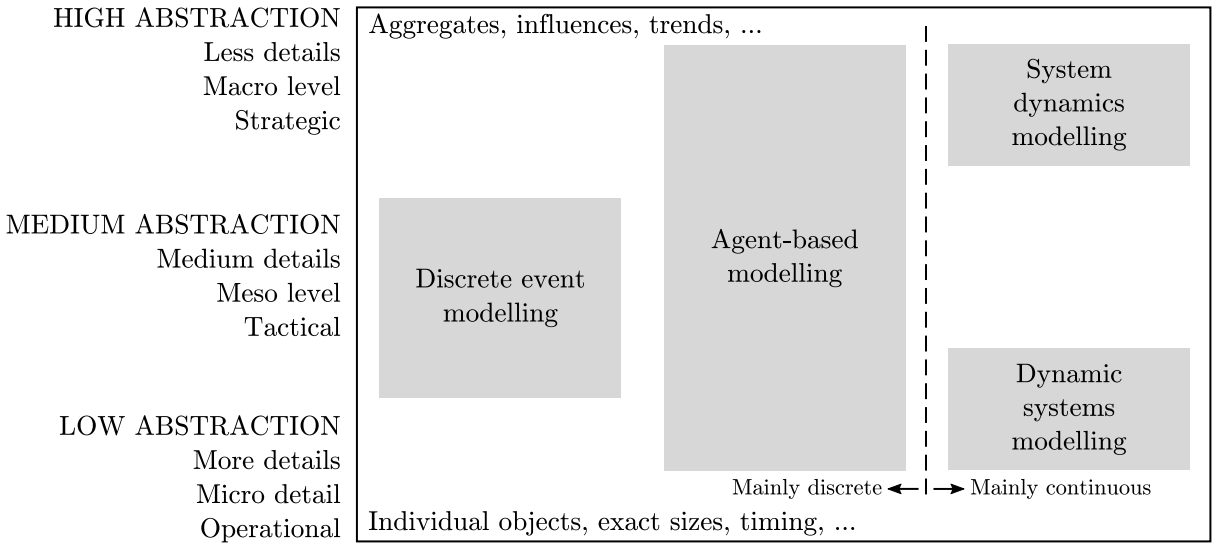


FIGURE 3.3: *High, medium and low levels of abstraction of simulation problems in conjunction with the four main paradigms of simulation models. Adapted from Borshchev et al. [21] and Grigoryev [56].*

discrete events), a list of discrete events that are expected to occur during the time period, a list of discrete events that have occurred since the beginning of the time period and a graph, statistical analysis or table of records of the events that occurred during the discrete event simulation process. Typical applications of discrete event simulation includes manufacturing lines, logistics, stress testing on materials and the evaluation of financial investments [114].

Agent-based modelling is defined as a decentralised, individual-centric modelling approach which identifies active entities and agents that posses individual behaviour traits in a specified environment [6]. Agent-based modelling, therefore, places focus on individual active components in a system. This provides an overview of the global behaviour of a system that emanates from the interactions between individuals in the system. Agent-based modelling techniques, therefore inspires dynamic interactions and communication between agents in a shared environment, with the purpose of evaluating their performance and to gain insights on the group's behaviour. An individual agent can perform functions that range from basic reactive functions such as "if-then" statements, to more complex and influential functions such as "belief-desire-intention" (BDI)[1] functions [1]. Traditional modelling approaches such as system dynamics modelling or discrete event modelling treat individual components of a system as aggregates or averaged quantities, whereas agent-based modelling is not limited in this sense [6]. Agents typically resemble people, families, vehicles, companies or products that are relevant to some system and that have individual connections (or relationships) with each other. By analysing the interactions between the various agents in the system, the global simulation model may draw overall conclusions or identify trends based on their behaviour. Typical examples of agent-based behaviour include how the decisions and actions of individuals may have an effect on the decisions and actions of their peers, or how the availability of individual aircraft may be dependant on rigid maintenance scheduling [6]. Agent-based modelling has a variety of level of abstraction from low to high, depending on the individual problem.

System dynamics modelling is a highly abstract form of modelling and is typically used in strategic models that are used over a prolonged period to represent characteristics of discrete

[1]Belief-desire-intention functions refer to the qualities an agent may poses to have a set of beliefs with respect to its environment, or a set of desires it wishes to maintain, or the execution of intentional actions that work towards some goal the agent wishes to achieve.

components such as people, events and goods in such a way that they do not possess individual qualities — the fine details that agent-based modelling systems include are ignored [6] and, therefore, has a high level of abstraction. This paradigm is typically used in complex systems where long term side-effects may be discovered via speed learning techniques and modelling approaches in a short time period [6]. System dynamics are capable of simulating cause-and-effect such as the effect of advertising strategies on brand perception by individuals, which enables the user to analyse the effect of the strategy used on the target market through a simulated analyses, rather than waiting a prolonged period of time in order to observe the effects in real-time. A typical example of system dynamics applications include the marketing plan of a telephone network company in order to analyse the probable success of a new telephone contract idea, without the consideration of individual customer interactions and influences.

Dynamic systems modelling is typically used in engineering disciplines such as mechanical, chemical and electrical engineering as a component in the design process [21]. Some of its first application areas included aerospace and military applications that include the use of technologically advanced and expensive systems. Thereafter, it was also applied in areas such as robotics and transportation systems, all of which rely on space and time variables [128]. This type of modelling is not widely used since it is usually applied to a very specific area of expertise that requires extended and customised disciplines and can, therefore, seldom be used in other applications due to the level of intricate detail it requires [128]. These models typically contain a substantial amount of state variables and algebraic differential equations that change discretely or continuously over time and are used to describe the model and provide some kind of input to a feedback control system.

3.4 Modelling elements

A simulation model is developed using various elements that are required in order to closely replicate a real-world system and subsequently create a so-called *digital twin* for the system. These elements typically include a *model*, various *events*, a variety of *entities*, various *activities* and a number of *variables*. Each of these are described in further detail below.

Variable. A *variable* provides information that describes a small aspect of a system. Entities are allowed to access and change these variables when events occur. A *state variable* describes the state of the system at any given point in time with sufficient detail. Discrete-event system state variables only change in increments at specified points in time (*i.e.* when events occur), while continuous system state variables change continuously over time.

Model. A *model* is a representation of an existing system and describes the aim of the simulation. It typically contains an objective function describing what the purpose of the model is, constraints describing the boundaries of the objective function and some decision variables that determine certain outcomes. The constraints may be complex enough to provide an effective representation of the real-world system, however, not too complex that the system imitation is too complicated and expensive to develop.

Event. An *event* refers to an incident that alters the state of the system, for example when a customer arrives at a service point. There are two types of events, namely internal and external events. *Internal events* (*i.e.* endogenous events) refer to events that occur within the boundaries of the simulation model, for example the arrival of an agent at a certain point in the simulation, whereas *external events* (*i.e.* exogenous events) refer to events that occur outside the boundaries of the simulation system, but still have an influence on

the overall system, for example a message that is sent in order to prompt a certain action to be performed.

Entity. *Entities* are elements of interest within the model that are defined discretely. They are dynamic objects that may move through the system, change their status, and they affect (or are affected by) other entities. Entities have custom assigned *attributes* that describe them and may be different from entity to entity. Entities compete for the "ownership" of various *resources*, allowing the entity to use or release these resources at any time during the simulation run. An example of an entity is an aeroplane waiting for access to the runway, or parts in a production line that have to be assembled.

Activity. An *activity* is a series of consecutive tasks that are performed with a known duration even before the simulation commences, which allows for time to progress. Thus, when an activity starts, its end time can already be established. The duration of an activity may be a specified discrete value following a distribution, for example a normal or triangular distribution, an input value determined by the user or from a data source, or a computational value based on an event that occurred. On the other hand, a *delay* is an unknown duration of an activity based on the consequences of events or system conditions. An example of a delay is when an entity is required to wait for a service or resource. The start and end of an activity or delay are events in the system that prompts something to happen.

3.5 Advantages and disadvantages of using simulation

As computers are evolving over time, users are provided with increased computing power, accuracy, faster computing ability and more user-friendly interfaces. Businesses are, therefore, more inclined to use simulation modelling in the problems that they have to solve. Although simulation modelling proves to provide high-quality results, there are a number of advantages and disadvantages associated with simulation modelling. In this section, the advantages and disadvantages of simulation modelling are discussed briefly.

3.5.1 Advantages associated with simulation modelling

Simulation modelling has many beneficial characteristics that makes it an attractive solution approach. One of the many benefits associated with simulation modelling is that a conceptual system which does not yet exist in practice may be tested extensively. This allows for an analyst to determine the optimal setup, operation, resource allocation, as well as identification and removal of problem areas of such a system without physically building and implementing it in a real-world context. It also reduces unnecessary correction costs that may be incurred if a model is not tested before implementation [9].

Another advantage that Banks [9] highlights, is that investigation time of various scenarios and their effects on a system can be manipulated by making use of simulation modelling. In a simulation environment, the running time of the model may be compressed or slowed down in order to observe intricate procedures or quickly skip over tedious procedures [86]. The analyst can also experiment with adding extra activities [86], such as adding an extra service teller in a bank or an extra shift for a workforce. The effect of such changes can be assessed in a short time frame at no additional costs.

Simulation reports tend to answer frequently asked questions by imitating the actual system and by examining all operations intricately in order to determine the reasons for certain occurrences and outputs. Banks [9] also acknowledges that managers are often interested to know why certain outputs occur as they do, but it is impossible for managers to investigate the system in such fine detail. Simulation modelling enables the analyst to keep track of all aspects, entities, and activities in its entirety and the reports obtained from the simulation reports may be used to address questions posed by managers.

A next advantage associated with simulation modelling is that once a model has been developed, the analyst may explore various scenarios with respect to operating procedures and other modifications of the model [8]. The effects of these modifications may then be observed by analysing the system output, without disrupting the real-life system, or incurring additional costs [86].

In modern times, factory floors and service organisations are (in some cases) so complex that it is impossible to take into account all the various interactions and communications that take place within these organisations at any given moment. The advantage of using a simulation model here is that it simplifies the operation of these complex systems by diagnosing problems and providing valuable insights on each variable and the effect that it has on the overall system [8]. By using a simulation model, a better understanding of the system and its working is obtained.

Furthermore, simulation modelling has the ability to identify components in a system which cause delays which may lead to bottlenecks in the system [9]. At the same time, simulation modelling also provides the opportunity to eliminate these bottlenecks, by altering and analysing the system variables.

A common misconception exists that people are able to simply think about a system and document it comprehensively and in complex reports and that by doing so, the system is deemed valid in a working condition and implementable. In reality, however, if a system is not tested thoroughly before implementation, a large possibility exists of the system being unsuccessful when implemented due to unforeseen human errors. Following a simulation modelling approach on a system aids the user in providing him/her with a better understanding of how the system actually operates, rather than depending on the perceived ideas of a human on how a system operates.

A simulation model also provides the opportunity to visualise a system and how it works through animation [86]. The system can be replicated visually in order to eliminate unanticipated design flaws as well as unpractical designs that appear good on paper or in theory, but are unpractical in reality. Visualisation of a system also allows for easy communication between individuals who have to understand the working of the system [9].

Moreover, research has shown that by taking preventative measures and developing a simulation study of a system, the user will save on many additional costs that are incurred by process redesign in the case where actual changes are made to the system without testing. Simulation modelling is, therefore, considered a good investment in order to save unnecessary costs over time and to prevent errors.

Specified requirements can be simulated with ease in order to observe the capabilities of a machine or production line, without physically testing them and wasting valuable resources [86]. In this sense, training can also be provided to teams in order for them to understand their role in an operation. Typical training procedures are discussed in further detail in §4.4.3.

Finally, simulation modelling may serve as a tool to reach consensus amongst various parties [9]. By repeatedly testing model outcomes and analysing results obtained through many simulation runs, the model's trustworthiness and effectiveness is confirmed by providing results and a

visualisation of what may be expected in the future. This could ultimately serve as a decision support tool to determine whether a project should be considered for implementation or not.

3.5.2 Disadvantages associated with simulation modelling

Even though simulation is considered a good solution methodology for solving problems, it does involve some disadvantages. The main disadvantages associated with simulation modelling are discussed in the remainder of this section.

Many hardware and software systems of the 21st century require that users undergo sufficient training to be able to use the software. Training involves some sort of cost and it can be quite time-consuming and expensive if executed effectively [8].

In most cases, the variables of a simulation model consist of a collection of random variables. It is, therefore, acceptable to assume that the output will also assume random variables. This could complicate the output-analysis of the simulation and subsequently make results difficult to interpret [8, 86]. Lately, however, suppliers of simulation software packages have developed output-analysis packages that significantly reduces the computational requirements of the user and is able to assist in analysing the results of the simulation by providing users with sensible information with respect to the model output.

Another disadvantage of a simulation model is that the construction and design of a simulation model is time-consuming and requires good quality data on the problem that is solved [8]. Good quality data are, however, not always readily available which may compromise the results obtained by the model. It can be very time-consuming to collect the correct data and many resources such as developers and data analysts may be required. Therefore, even though the cost of testing a model before implementation may reduce the design costs, the costs of collecting data and building the model can be high. Advancements in technology and improving hardware equipment that provides faster solutions may, however, be used to solve this problem [8]. Furthermore, most of the existing complex simulation packages have updated versions available that are adapted according to problems experienced with previous software versions, for example a user will report a problem (s)he is experiencing and the software developers will address the problem and release a newer version of the software in which the problem does not oocur. Some software packages are also available in different user-competency levels (*i.e.* beginner to advanced versions) in order to provide the user with a working environment that he/she is comfortable and sufficient in.

Finally, simulation models may be used inappropriately [8]. In some instances, a problem can easily be solved analytically or an analytical method may be preferred to solve the problem, but a simulation modelling approach is followed [86]. If this happens, the results may be substandard solutions that were picked according to a user's preference, rather than providing optimal results when an analytical approach be followed. On the other hand, it is very seldom that a real-world problem is simple enough to solve analytically and, hence, a simulation modelling approach seems natural to solve the problem.

3.6 Developing a simulation model

When developing a simulation model, a number of general steps may be used to achieve this. Banks [8, 9], Law and Kennon [78], and Maria [86] discuss twelve steps that may be followed to

develop a simulation model. The order in which these steps should be implemented is illustrated graphically in a flow diagram in Figure 3.4.

1. *Problem formulation.* In order to solve a problem correctly, it is imperative that the problem formulation is comprehensive, detailed and complete in order to avoid confusion with respect to achieving the end goal. It is also important that all the demands of the client are met and, therefore, clearly stated in the problem formulation [8]. It is also recommended that the analyst formulate some assumptions with respect to the problem that should be certified by the client in order to ensure (s)he agrees on the formulation strategy.

2. *Project planning and scope.* The objectives of a problem give an indication of the questions that require answers throughout the simulation study, such as "how many resources are required to perform the tasks effectively?" or "where are the bottlenecks in the system located, and how can they be resolved?". The analyst and client have to decide on the various scenarios of the problem that require further investigation, as well as which decision criteria have to be considered. The project plan should include a detailed description of the data that is available, as well as which data is required in order to successfully solve the problem and validate the solution obtained. The project plan should be accompanied with a projected time line plan in order to effectively schedule work tasks and adhere to the deadlines. Furthermore, a resource and cost plan should be compiled in order to determine the financial aid and labour requirements for the project over its duration.

3. *Model conceptualisation.* The simulation model being developed is an abstracted version of a real-world problem and is represented by using mathematical and logical relationships that represent the layout and components of the system. These abstract features include the system entities, relevant activities, events, attributes, variables, resources and the relationships between these various features. It is important to find a balance between the amount of complexity and elementary components that are incorporated into the simulation model, since *too little* complexity may result in a loss of important information required to obtain sensible results, while *too much* complexity requires more time, resources, longer simulation runs, and is also more likely to contain errors. The proposed approach for modelling a complex system is to start with simple components (*i.e.* with as little detail as possible) and add complexity in small increments until the model represents the real-world situation as realistically as possible.

4. *Data collection.* Data collection is very important for input analysis purposes, as well as model validation, because the quality of the model results relies on the quality of the input data received by the model. Data collection has typically been executed by the client over a substantial time period. In many cases, the data received from the client is in the wrong format or not documented correctly and completely. Therefore, time has to be set aside for data cleaning and analysis in order to ensure the input data is in the correct format and contains the necessary information for the simulation study to run correctly. It is also recommended that the model building procedure, as well as the data collection procedure are executed concurrently in order to ensure that the correct data is available for each stage of the simulation model.

5. *Model translation.* During the model translation phase, the conceptualised model from Step 3 is converted into an operational model in the form of the computer-recognised language required by the software suite that is used to solve the problem. It is preferable to use programming languages that are specifically designed with the purpose of solving

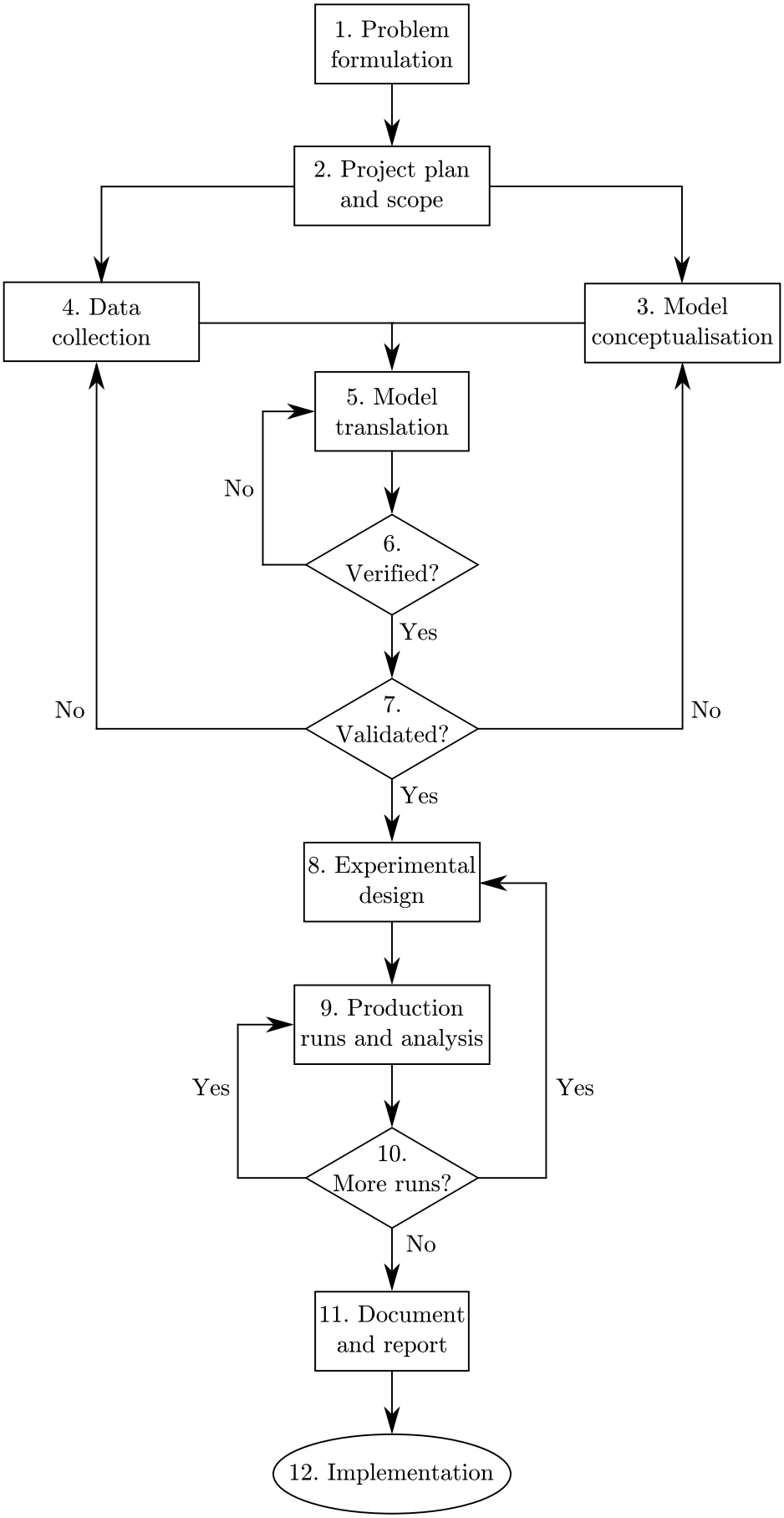


FIGURE 3.4: *Twelve steps involved in building a simulation model [9, 119].*

problems similar to the problem at hand (*i.e.* specialised software), rather than commonly available programming languages. The main reason behind this is that specialised software may reduce computational time and may increase model efficiency, whereas standard software may not always provide these benefits and would simply solve the problem in a less impressive and less-detailed manner.

6. *Verification.* Verification is defined as the process of establishing whether the operational logic of the simulated model is working correctly, and whether it was tested by debugging the software. It is widely recommended that verification take place concurrently throughout the model development process, rather than only performing verification once the model is finished. This reduces the time required to verify and the redesign time of the model, if required. Large operational errors may also be prevented by performing verification.

7. *Validation.* Validation is defined as the process of establishing whether the simulated model accurately represents the real-world system that is simulated. This is executed by comparing the results obtained from model output with the actual results from the real-world system. Unfortunately, a real-world system is not always available to compare the model results with and, therefore, other methods of validation may be used. Bekker [10] mentions five ways in which the validity of a simulation model may be tested. These methods include,

   (a) *continuity* – subtly adjusting input parameters that should only have similar effects on the output parameters and overall results,
   (b) *consistency* – similar runs of the model should provide similar results as outputs,
   (c) *degeneracy* – the model results should realistically reflect changes made to any of the input parameters,
   (d) *nonsensical conditions* – in the scenario where nonsensical data is provided as input parameters to the model, the model should recognise the absurdness and not provide nonsensical results as a result, but rather provide an error message, and
   (e) *extremes* – the model should be tested at the extreme scales of the model parameters (*i.e.* minimum and maximum limits).

8. *Experimental design.* In this phase of the simulation study it is important to test the working model in different scenarios to observe the effect of changing parameters on the outputs. Here, the performance measures of the system are determined. For each variation of the model run that is executed, decisions have to be made in terms of how many simulation runs have to be executed, the time duration of each run, as well as how each run is initialised.

9. *Production runs and analysis.* Production runs and the analysis thereof are used to determine system specific performance measures that can be used to interpret results in terms of their statistical significance. This enables the analyst to compare different configurations of various runs and to establish which configuration aligns with the problem objectives, or to obtain an improved understanding of the possible scenarios the real-world system may experience.

10. *More runs.* More simulation runs on different scenarios are directly related to a higher confidence when interpreting the results according to the performance measures. Therefore, if an analyst is unsure about some of the model outcomes, more simulation run scenarios should be executed in order to eliminate the level of uncertainty experienced and to improve on the accuracy of predicting the real-world system.

11. *Documentation and reporting.* It is important to effectively document the entire simulation model since the same model or a similar one may have to be constructed or updated in the future by other analysts or in other applications. Documentation also provides a source of reference for the client to consult when making decisions based on the analysis. Two aspects of the model requires detailed documentation *i.e.* the *program* that was built to implement the system along with the model features, as well as the *results* achieved through the simulation study. This may result in a significant amount of time and effort being saved when future analysts are trying to understand the working of the model, and also ensure that the results achieved are in line with what was originally intended and expected. Additionally, other details of the system such as model requirements, model animations, model tests and sporadic model reports are essential in order to provide an accurate representation of the simulation model to external and internal associates.

12. *Model implementation.* Implementation is more likely to be successful if the previous eleven steps where executed successfully and effectively, and if the client and analyst both fulfilled their respective roles in the development process with merit. In such a case, the model may be implemented successfully and the decision maker may be able to gain valuable information from the model on which to base their decisions.

3.7 The difference between optimisation and simulation

The two terms *optimisation* and *simulation* are often misused, confused or interchanged, even though the two are completely different concepts and should be used accordingly. According to Lee [80], optimisation refers to a scientific approach which is taken during a decision making process in order to find an optimal or most efficient solution, while adhering to a set of constraints that relate to the problem objective. The objective is typically in the form of a mathematical expression of functions that comprise a number of variables and should typically be maximised or minimised. The constraints enforced in the problem are typically mathematical expressions which represent the limitation of resources that are involved in the problem. Lee [80], on the other end, describes simulation as the evaluation of a large number of different realistic scenarios under flexible conditions. Simulation may be used to support decision making by evaluating different options and drawing conclusions or identifying trends. It is, therefore, required that the decision makers involved have sufficient knowledge of the field which is being explored. Furthermore, simulation is commonly used to provide decision makers with a virtual experience of real world scenarios and to provide insights into the specific behavioural traits of a system [80]. Some of the advantages and disadvantages associated with optimisation and simulation are listed in Table 3.1 and are discussed below.

According to Lee [80], some advantages associated with optimisation modelling include that it produces high quality, analytical solutions and that it may be tactically and strategically applied to a wide range of application areas. Some disadvantages associated with optimisation modelling include that it tends to oversimplify the problem and, therefore, provides a less realistic solution since uncertainty is not included in the problem formulation. Some advantages associated with simulation modelling include that it produces highly practical solutions including minimal assumptions and it includes uncertainty which, therefore, qualifies it as a long term solution. Some disadvantages associated with simulation modelling include that it is difficult to obtain high quality solutions with little effort — high quality solutions are only achieved through investing time and resources which, in turn, may make it a very high cost procedure.

	Optimisation	Simulation
Advantages	High quality analytic solutions	Highly practical with minimal assumptions
	Tactical and strategic solutions for many application areas	Includes uncertainty and entails a long-term strategy
Disadvantages	Tends to oversimplify the problem	Difficult to obtain high-quality solutions
	Less effective if the problem involves uncertainty	High costs

TABLE 3.1: *Summary of some of the advantages and disadvantages associated with optimisation and simulation, respectively. Adapted from Lee [80].*

There does, however, exist an approach which combines the two solution methods into a single approach and is known as *simulation-based optimisation* [16]. Simulation-based optimisation integrates the techniques used in optimisation into the analysis approach of simulation. The approach, therefore, allows a problem to be modelled mathematically first (*i.e.* optimisation modelling) and simulated thereafter in order to incorporate behavioural traits. In this case, however, the objective function becomes difficult to determine and expensive to evaluate due to the complexity which is associated with simulation analysis [99]. The primary aim of a pure simulation analysis approach is to evaluate the effect of variations on input parameters which are provided to the system, however, in most cases it is beneficial to know what the optimal value for the input parameters is and, in this case simulation-based optimisation is preferred. The optimal value for the input parameters may then be established by running various simulation experiments with different input parameter values [26].

In conclusion, it cannot be said that optimisation is a better solution approach than simulation, or *vice versa*. It may rather be assumed that optimisation is better suited as a solution approach for some problems, while simulation is a better solution approach for other problems and, in some cases, the two may be used as a combined solution approach. When optimisation and simulation are used as a combined solution approach, however, the problem must first be optimised and then simulated and the combination may result in a few difficulties when attempting to accommodate the advantages and disadvantages of both solution approaches.

3.8 Chapter summary

This chapter provided a sufficient foundation for the understanding of simulation modelling from the operations research literature. The chapter opened in §3.1 with an introduction towards simulation modelling where a definition and some basic history on simulation in general was provided. Thereafter, in §3.2 the various paradigms of simulation modelling were discussed, with specific focus on static versus dynamic simulation models, deterministic versus stochastic simulation models and continuous versus discrete simulation models. Furthermore, the different levels of abstraction according to which models may be developed including high, medium and low abstraction and modelling methods such as discrete event modelling, agent-based modelling, system dynamics modelling and dynamic systems modelling were described in §3.3. A simulation model is typically developed using various elements in order to replicate a real-world problem and

each of these elements were discussed in §3.4. Next, in §3.5 the advantages and disadvantages associated with simulation modelling were highlighted. In in §3.6, the twelve steps that may be followed in order to effectively develop a simulation model were mentioned and, lastly, there was a discussion on the difference between optimisation and simulation in §3.7. The chapter finally closed with a brief summary on the chapter contents.

CHAPTER 4

Decision support system literature study

Contents

The aim in this chapter is to provide a sufficient foundation for the understanding of DSSs in the literature. Frameworks in general are briefly mentioned in §4.1. Thereafter, decision support frameworks and their components are discussed in extensive detail in §4.2. Section 4.2.1 focuses on the general structure of a DSS and its components and each component is described in detail along with the component's accompanying terminologies. This section also includes the benefits associated with designing a good database, the five main types of database management systems, and guidelines for designing a GUI. Section §4.2.2 focuses on selecting appropriate software for developing a DSS and also provides some guidelines that may be followed when developing a DSS, while section §4.3 discusses the systems development life cycle that is deemed relevant when developing a DSS. Sections 4.4.1–4.4.3 discuss methods that may be used to ensure quality in DSS development and also suggest techniques that may be used to test, verify and validate the development of a DSS, as well as how to provide training to the end-users. Thereafter, sections 4.4.4 and 4.4.5 discuss the implementation and maintenance of a DSS, respectively. The chapter finally closes with a brief chapter summary of the chapter contents in §4.5.

4.1 Frameworks in general

A *framework* may be defined in a broad sense as a skeleton or architecture of processes that are interlinked and grouped together to ultimately serve as a supporting mechanism in order to

achieve some predefined objective. Frameworks are typically designed in such a way so as to aid users in specific decision making processes by including various unknown factors that the user should take into account before solving unstructured problems. They offer simple collections of different concepts grouped together in a single setting (or as a construct) that lays out key factors, variables and relationships of a specific system or model. These concepts are then used as a mechanism to merge multidisciplinary bodies of knowledge into a single field of study with the purpose of achieving a common goal. Moreover, they aim to provide an interpretive angle on a problem, as well as an explanation of the intentions behind a solution strategy. Frameworks, therefore, aid in the creation of an improved understanding of the problem at hand, rather than simply providing a general analysis of the problem. Frameworks require qualitative analysis and case study analysis in order to provide accurate insights into the problem as well as the possible outcomes.

The concept of a framework serves as a basis to map out the development and design of DSS with the goal of making the DSS logical and simple to implement and use. It also provides insight into the ultimate purpose of the proposed DSS and how a user would be able to effectively use and implement it.

4.2 Decision support systems

DSSs have evolved greatly over the past few decades due to information system technology becoming progressively more advanced. They inconspicuously diffuse into the spectrum of computer-based tools that are regularly encountered in modern day office environments. DSSs serve the purpose of presenting data and results returned from model frameworks to multiple groups of people (typically situated in an organisation) in such a way that it is easy to interpret or understand by these individuals.

A number of definitions for DSS exist in the literature. These definitions depend on the circumstances surrounding the system that has to be modelled. Keen [68] formulates a general definition of a DSS which states that a DSS may be interpreted as the use and application of appropriate computer-based technologies to aid the improvement of effective top-level decision making within the context of semi-structured problems. Kendall and Kendall [69] define a DSS as a formally defined and centrally controlled store of data intended for use in various different application areas. In essence, a DSS may be seen as a *support system* in the sense that it should help a user negotiate through a series of decisions and processes in order to successfully perform a specific task. This is due to the final version of the system only emerging through a flexible process of design and redesign.

The design of a DSS typically requires multiple iterations. In some cases, the designer and/or user have insufficient knowledge of the problem at the initial stage of the design process and therefore have to iteratively redefine procedures and requirements as the complexity of the problem increases. A DSS is used to stimulate learning and provides additional insights to the problem which subsequently provides an opportunity for new applications and updates to the original DSS. The design of a DSS may be seen as a two-way street where the DSS forms the user and the user also forms the DSS. Therefore, the design requires ongoing interaction throughout the entire design process. Furthermore, it is important to include the skill level of the end-user when designing the DSS, since the DSS can only be of value to the user if (s)he is comfortable using it.

According to Keen [67], there are three cornerstones in developing a DSS, *i.e* the user, the builder and the system. Figure 4.1 illustrates the interactive processes between these three

cornerstones. The direction of the arrows in the figure indicate the direction of influence, for example, the system teaches the user, but the user can personalise the system. Similarly, the builder improves the functionality of the system, but the advancement in the system functionality pressures the builder to continuously improve his/her skill level. Finally, the builder facilitates the implementation on the user side, but the user may communicate his/her preferences to the builder via a middle-out approach which refers to the fact that the builder has to learn from the user and that the user has an input in the design process [67].

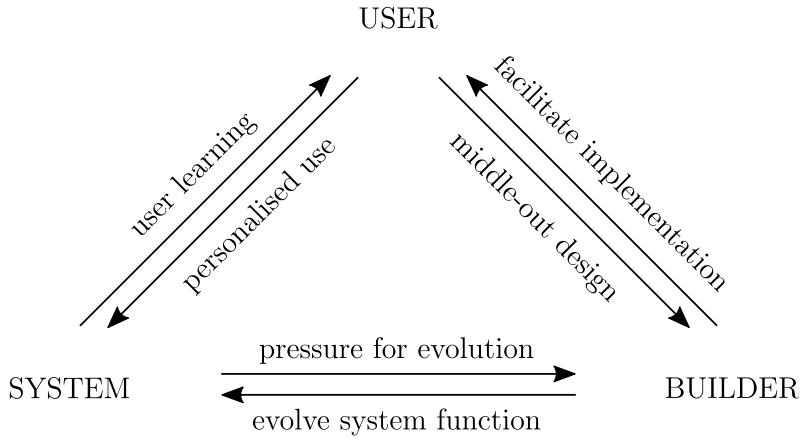


FIGURE 4.1: *Three important cornerstones to include in the development of an adaptive framework for a DSS [67].*

4.2.1 DSS structure

According to Management Study HQ [85] a DSS typically contains three essential parts, known as a *database*, a *GUI* and a *model base*. Figure 4.2 illustrates a general structure of a DSS and was adapted from Deogun [39]. This structure contains various integral components of a DSS and may be adjusted and customised as deemed necessary by the developer in order to include various aspects and intricacies that are required to solve and support the decision making of a specific problem. The DSS is constructed by using various components that each serve a specific purpose in the overall working of the DSS. The remainder of this section is dedicated to briefly discuss each of these components along with their respective accommodating terminologies.

The first component is the *central intelligent unit* denoted by (1) in the figure and introduces aspects such as logic, linguistics and artificial intelligence which enables the system to perform reasoning, problem solving and other learning techniques. This component is therefore linked to the second component, the *knowledge base* denoted by (2) in the figure. The knowledge base enables the system to reason effectively and operate on a cognitive level. Moreover, the central intelligence unit processes problems in terms of the goals and the requirements specified by the organisation. The analysis of these processes can be achieved in a *work space* environment which involves a team that analyses the problem. The knowledge base is important in this regard since it should help the user in recognising a problem, dividing it into subproblems and describing how each of these subproblems may be solved effectively.

The third component is the *model base management system* denoted by (3) in the figure and typically forms the heart of any DSS. This component contains the actual mathematical model(s) that are employed to solve the problem. This component is therefore expected to involve a significant amount of data processing and is therefore linked to the fourth component *i.e.* the

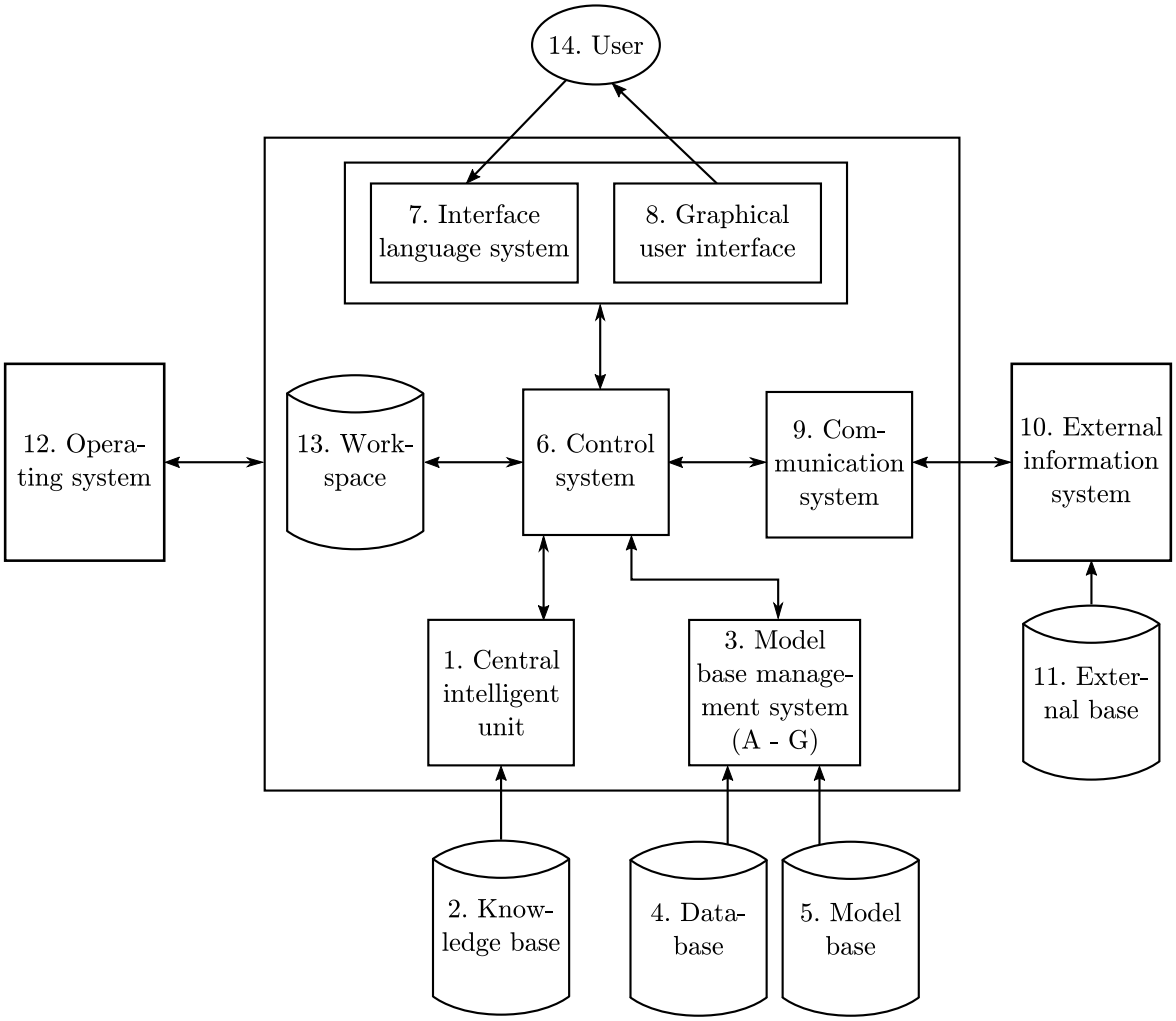


FIGURE 4.2: *General structure of a DSS, as adapted from Deogun [39].*

database denoted by (4) in the figure. The model base management system considers the fact that models may contain submodels and therefore stores all models used in the DSS in a *model base* denoted by (5) in the figure. The third component is therefore considered to be quite complex. Malhotra *et al.* [84], however, suggest a general DSS framework involving the use of multiple submodels that provide different output results. The approach provided by Malhorta *et al.* [84] presents the opportunity for the user to then choose an outcome from a number of possible alternatives which appeal to him/her the most. The DSS for multiple submodels as formulated by Malhorta *et al.* [84] may therefore be used as an extension on component (3) in the DSS proposed by Deogun [39] by incorporating components (A)–(G) as portrayed in Figure 4.3.

The framework presented by Malhorta *et al.* [84] contains seven components which work in conjunction with one another. The flow of information between these components are illustrated by means of solid and dotted lines and the direction in which the information flows are indicated by arrows. The solid lines represent information that is required to analyse observations throughout the main process flow, whereas the dotted lines represent information where census was not initially agreed upon and, therefore, the information has to be reconsidered and reconstructed by the parties involved.

Each model classification type (C.1–C.n) considers two variables, namely: *dependent* and *independent* variables, where dependent variables are predefined, and independent variables are

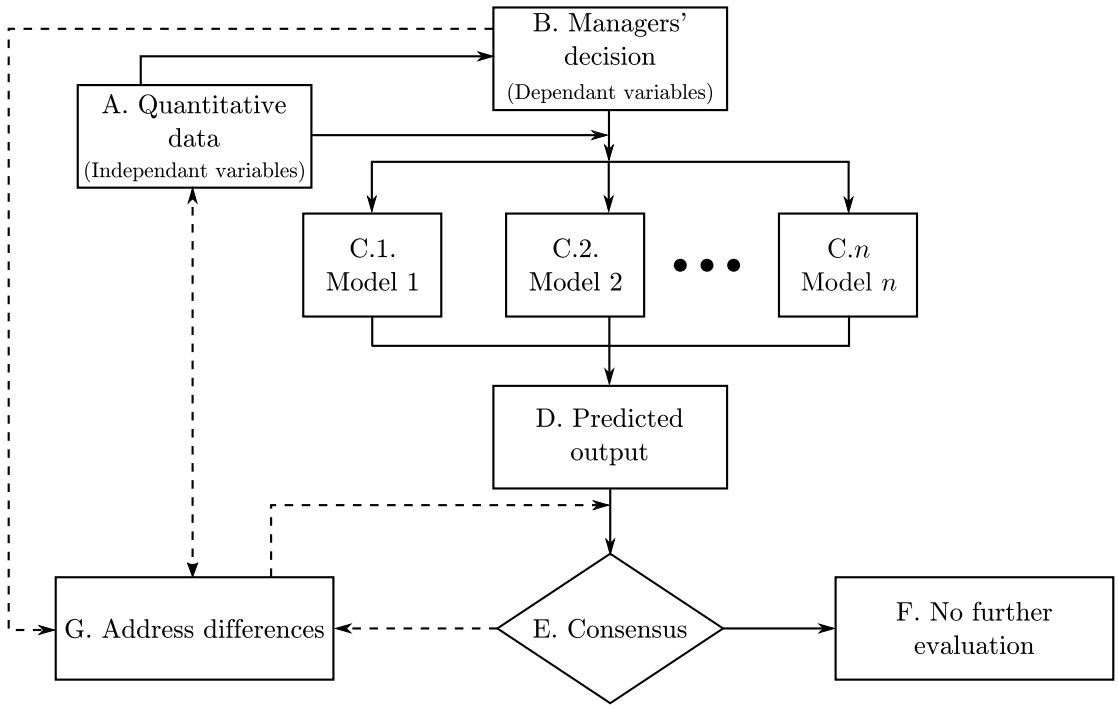


FIGURE 4.3: *A general DSS model framework containing its components and the flow of information between the components, as adapted from Malhotra [84].*

acquired from historical data and performance indicators. The processes in the DSS are limited by reading in qualitative data (represented by the square labelled (A)), which the system requires. These data are also referred to as independent variables to the system in the sense that they are acquired from other sources that are not dependent on the results of the model or on the managers' inputs. This information is then relayed to select variables that are dependent in the sense that the managers have some input with regards to these variables. After the dependent variables have been identified, all the data (consisting of dependant and independent variables) are relayed to the model component (C.1–C.n) to solve the problem. The models are calibrated based upon the input data received. Each model is executed and the results thereof is relayed to the output component represented by the square labelled (D). Each model output is reviewed and judged upon whether or not there is consensus on the models' predicted output (E) (*i.e.* do the various outputs correlate and do the outputs confirm the managers' initial judgement). If the managers agree that the output is usable, consensus is reached and no further evaluation is required (F). If, however, there exists disagreement between the managers the managers have to reconsider the output and address the differences (G). This review process to address the differences may require additional information to be introduced in order to make informed decisions and ultimately reattempt to reach consensus. The DSS framework in Figure 4.3 has the advantage of providing clear consensus on outputs that confirm managerial decisions and confidence, as well as highlighting outputs that require further investigation and possible refinement. This allows the managers to focus their attention on particular areas of the data instead of reconsidering an entire set of data. This all forms part of component (3) in Figure 4.2.

The database denoted by (4) in Figure 4.2 is a central component in a DSS. The database is employed to effectively store data in an easily accessible and structural format so that it may be accessed to perform various analyses or to obtain specific information regarding objects or concepts. Furthermore, a database may be seen as a structured collection of various data that may be used to aid the activities of a specific organisation [145]. Examples of such data

include system characteristics, system components, parameters and variables. Databases comprise various properties depending on the nature of the database. One overarching property is to portray real-world information through the structured presentation of data elements. Furthermore, databases require consistency and logic in terms of the structure of the data that is stored and should therefore be designed, built and populated with relevant data that serves to represent a common purpose. Each data item should be stored in a specified data field and a combination of these fields are used to create a table of relevant information. The performance (*i.e.* the success of a DSS) of a DSS depends on the design of the database. Watt and Eng [145] describe the benefits associated with designing such a database and they advise to include the following aspects when designing a database:

Data redundancy. The database should not have any redundancy. The developer of the database should ensure that no duplicate elements are created in the data.

Concurrency. The database should allow concurrency, which involves providing access to the data for multiple users simultaneously.

Rectitude. The database should be rectitude *i.e.* not to allow any flawed entries in the database.

Security. The database should be secure *i.e.* involve restrictions to access and manage data amongst users.

A database commonly requires a data manager as an internal component, which has the ability to manage, process and store information easily and fast. A number of different types of database management systems exist in the literature. The nature of the database management systems depends on the manner in which the data is stored in the database and managed in the database manager. Obbayi [100] mentions that there are five main types of database management systems, which includes *relational* database management systems, *flat-file based* database management systems, *hierarchical* database management systems, *network* database management systems, and *object-oriented* database management systems [100].

Relational database management systems. They are widely used due to the low level of complexity required to implement them. These database management systems are typically used when large complex systems of data are involved. They focus on normalising data in rows and columns of these data sets according to their characteristics. These databases are usually designed in conjunction with *entity-relationship diagrams*[1] (ERDs). These diagrams consider all components in the system as entities with relationships, where the relationships may either be one-to-one, one-to-many or many-to-many. These relationships describe the number of associations that exists between two entities, for example

- *one-to-one* relationship is a person and a passport, because a person may only have one passport and a passport may only be associated with one person,
- *one-to-many* relationship is a museum and original works of art, because a single museum may house many original works of art, but one original work of art cannot be exhibited in many museums,
- *many-to-many* relationship is books and authors, because a book can have many authors and an author can also have many books.

[1] An entity-relationship diagram is a technique used to present data graphically by illustrating the entities in an information system, as well as their direct relationships [130].

The relationship of an entity is determined and changed by using a *structured query language*[2] (SQL) [100]. In using this software, the data may be stored in various access tables each having a key field as an identifier. Each identifier refers to a specific column or row in the data [104, 123]. Relational database management systems are, however, considered to be slightly less efficient than the other four models when considering processing time.

Flat-file database management systems. They are considered the most straightforward of the five database management systems and are typically structured in the form of human-readable text documents or binary files [123]. These database management systems are considered ideal for independent applications, such as internal computer applications that are used to store data related to the configuration files. These database management systems may only be used when no missing values are present in the data — all the entries in the database should have a value. If one or more entries in the database are missing, the file is not executable. An example of a flat-file database management system is Microsoft Excel which uses *comma separated value*[3] (CSV) files [100]. These files use commas to separate data into columns and rows (in the form similar to that of a matrix), and it is assumed that all data in the same column or row of a data table is of the same type.

Hierarchical database management systems. They function according to a parent-child tree structure and generally involves a one-to-many relationship-type [100]. A parent-child tree typically has a superior node (*i.e.* the parent node), which will generally have zero or more child nodes, which are nodes that involve data that is typically able to describe some aspect of the parent node. These systems are able to store data (*i.e.* parent nodes) in conjunction with their character attributes (*i.e.* child nodes) in a nested format that takes the shape of a tree diagram [104]. One of the disadvantages associated with these systems, however, is that when new fields or records are added to the tree structure to increase complexity, the entire database has to be redefined since a single entry may simply not be inserted anywhere in the tree [48]. This may result in a significant slower execution time. The most common method of storing data according to this model structure is by using *extensible markup language*[4] (XML) file formats [104].

Network database management systems. They are similar to the hierarchical system in several ways. Two similarities include that they both take on a nested form (*i.e* a tree diagram or a network diagram) and that they have child nodes that provide characteristic information on the parent nodes [100]. The main difference between the network system and the hierarchical system is that the network system has the ability to accommodate many-to-many parent-child relationships. Thus, instead of viewing the database as a tree structure, it may rather be viewed as an interlinked cobweb in the sense that child nodes may be connected to multiple parent nodes and parent nodes may be connected to multiple child nodes [104]. The network system contains various records and data sets that may be manipulated using SQL software. Although these models are considered flexible, the disadvantage associated with them is that the entire network of data entries has to be searched in order to identify a single record, which may be a very time consuming process.

[2] A structured query language is a standardised query language that is used to access and modify information (*i.e.* commands such as *insert*, *update* and *delete*) in a database [132].

[3] Comma separated value files contain the values of a table in a series of 7-bit binary number (ASCII) coded text lines in such a way that every column is separated by a comma from the next column, and every row begins in a new line [131].

[4] Extensible markup language is a universal language format that is used to represent and transfer structured data on the web or from one application to another [129].

Object-oriented database management systems. These systems are very closely related to object-oriented software suites such as `C++` [18] or `Java` [79] and make use of *pointers* to identify certain characteristics, attributes or features of a particular datum point [100]. Data are therefore not stored in relational tables but are rather stored in diverse, object-based data structures. These systems are designed to work in unison with the software suite and the applications therefore require less coding, the data modelling is more natural and the code is easier to maintain and adapt [104]. This type of database management system is a relatively new concept and has consequently not been used widely.

According to French [50], all databases are unique and, therefore no "one size fits all" database solution exists. An existing databases from the literature may, however, be used as a stepping stone to construct a unique database specific for a certain DSS and more specifically for a given model used in the DSS. The database that is constructed for a specific DSS should be problem-specific and the advantages and disadvantages of the database should be considered carefully.

The next component in Figure 4.2 is the *control system* denoted by (6) which integrates and coordinates the various components that are necessary to process data in order to support decision making. This component therefore converts the knowledge level to a procedural level in order to obtain a better understanding of the decision-making activities on an operational level.

The next two components are grouped together and comprise the *interface language system* which is denoted by (7) in the figure and the GUI which is denoted by (8) in the figure. The reason why these two components are grouped together is that they are inter-dependant. A GUI may be seen as the link between man and machine and creates a *human-computer interaction* (HCI) environment in the DSS that enables the *user* (which is denoted by (14) in the figure) to interact with the system and provide certain input to the DSS. This environment typically exists through the use of computer screens, user forms and GUIs. In most cases, the user is required to provide inputs to the system, while the GUI provides a platform to do so and also presents the results in a visual form of output that the user may interpret and analyse. Since *interaction* is such an important component of HCI, the GUI should fundamentally be user-friendly. In order to achieve this goal, Kendall and Kendall [69] suggest a few guidelines to follow when attempting to design a realistic system that embodies and represents the real system as accurately as possible. The first guideline is to include *task examination*, which is used to determine the *suitability* of a HCI in the environment of the user — the GUI must be relevant. The second guideline is to *identify* and address possible obstacles that users may encounter when attempting to carry out their assigned tasks. The third guideline is to ensure *user-friendliness* by ensuring that the HCI is useful in each situation, as well as to check whether the implemented technology will be easy to use and provide the appropriate feedback that users require. The fourth guideline is to *examine the usability* of the HCI by analysing the interaction between the user and the technology and how they perceive it, as well as what the user expects to receive or achieve through the HCI. The final guideline is to *design prototypes* in order to accommodate diverse users and ultimately improve their productivity.

A number of GUI types exist that may be implemented in DSSs. Examples of these GUIs include *natural-language* interfaces, *question-and-answer* interfaces, *menus* and *form-fill* interfaces. Natural-language interfaces involve verbal interaction with a user. They typically do not require special skills from the user since the user can interact with the system via a low-level complexity platform such as a voice-enabled interface. Examples of such interfaces include Siri from Apple Incorporated and Alexa from Amazon. Question-and-answer interfaces involve an interactive exchange between the system and the user and act on a prompt-and-respond basis. Here, a computer will typically display a question to the user on a screen and the user is allowed

to reply (or answer the question) by either typing an answer using a keyboard or clicking on an answer by using a computer mouse. The computer will then process the answer and respond accordingly. An example of a question-and-answer interface is the one of signing up for a social media account, for example Facebook and Instagram. The user is required to enter their credentials in a number of subject fields after which the user can successfully register their personal social media account. Menu interfaces involve options that a user may investigate and navigate, but limits the variety of information and responses a user can provide to a predefined set of options. These interfaces are typically represented in a nested form which tends to group interrelated objects together in order to present the user with only the options that are of interest to him/her. An example of a menu interface is the tab menu available at the top of the page in Microsoft Word [93] for editing a document by toggling between the various tabs and their respective menus. Finally, form-fill interfaces involve a form with various blank fields that have to be completed by the user with relevant information and present various fields of data items or parameters to the user that (s)he may use to customise the system. Blank fields are required to be filled in and are typically highlighted to the user in order to draw their attention. Input fields could also be restrained alphanumerically in order to restrict the input provided by the user and to keep the input data relevant. An example of the form-fill interface is web forms.

The implementation of GUIs depend on the expected HCI and outcome. It may be observed that all types of interfaces have two common components, namely *presentation language*, which represents the computer-to-human component of the interaction, and *action language*, which represents the human-to-computer component of the interaction. Kendall and Kendall [69] suggest a number of criteria that may be used when choosing and evaluating user interfaces [69]. The criteria includes choosing an effective HCI system, since users respond very well to systems that they find appealing, user-friendly and practical. In addition, it is necessary to consider the experience level that a user has (*i.e.* whether they have worked with the HCI before, or if it is their first encounter with the HCI). Furthermore, the training period for users should be short and acceptable for all users and everyone should feel capable of using the software without continuously referring to a help menu. Finally, the interface design should be seamless and contain no design related errors and, in the case where an error does occur, it should be easy to resolve. Wang and Tan [144] suggest six guidelines for the design of GUIs which are as follows.

User familiarity. In order for a user to be able to effectively and clearly perform their task, it is recommended that they are only provided with data and information that is familiar to them. Details related to algorithm and data structures should therefore typically not be made visible to them — they do not require this information to perform their duties and, hence it is not of importance to them and will not benefit the process.

Consistency. Research has shown that higher levels of HCI consistency results in more productive users [144]. Therefore, if the interfaces presented to the user conform to a certain layout, users can easily become familiar with the system and orientate themselves in the various areas of the HCI.

Minimise surprise. This guideline lies in unison with consistency in the sense that when a user provides certain inputs to the system, (s)he will expect certain outcomes from the system. The system should, therefore, be designed in such a way that user requirements are achieved and that the risk of providing incomparable results are minimised. For the same reason, system functions should be presented in a straightforward manner so that the user may easily navigate between features.

Recoverability. This involves identifying and rectifying faults in the system. Users are subject to human error and it is therefore necessary for the system to include features in the system

that are able to rectify such human errors. Examples of recoverability include auto-correct, restriction to alphanumeric values or check-pointing (a system's ability to retrace to a point where no faults were part of the system, for example when using the "Edit-undo" function in Microsoft software).

User guidance. It is important for a system to include user guides that are able to assist the user according to his/her level of expertise. This typically comes in the form of a *help guide* and is sometimes included as a built-in menu in systems that are easily accessible by the user.

User diversity. Systems should be developed in such a way that they are easily understood by their users. It may therefore be useful to have various user interfaces available ranging in different levels of complexity according to the user's level of knowledge of the system. A new user to a system for example, may be presented with a simple GUI, while a more experienced user may have the same GUI as basis, but may have added advanced features that (s)he can utilise when necessary.

Considering the six design guidelines described above, as well as the previously mentioned criteria for choosing and evaluating GUIs, it may be concluded that designing an effective and user-friendly GUI is an iterative process and that an adequate GUI will only be established after a number of prototype attempts.

The next component in the DSS in Figure 4.2 is the *communication system* which is denoted by (9) and equips the DSS with an environment where information may be transferred between the various parties involved in the system (*i.e.* in the case where one company has different branches across a country) and is therefore linked to the *external information system* which is denoted by (10) in the figure. The external information system stores any relevant information required for decision-making purposes. Therefore, the external information system is subsequently linked to various *external bases* which is denoted by (11) in the figure and provide access to this information.

Finally, the *operating system* denoted by (12) in the figure makes provision for several users to access the system simultaneously and, therefore, includes security systems, linkage systems and management systems that control the resources available in the system.

4.2.2 System software

When selecting a software suite to develop a DSS in, it is important to choose one that will best suite all the user-requirements. A number of such packages should be considered in order to determine what their respective capabilities are and whether it is applicable to the DSS that is being developed.

Kendall and Kendall [69] suggest a number of guidelines (in the form of questions) which developers of a DSS should consider when evaluating software that could be used for the implementation purposes of a DSS. The guidelines proposed by Kendall and Kendall [69] are as follows:

Performance effectiveness. Is the software able to perform all the tasks as requested? Does the software have adequate display screens? Does the software have adequate capacity to meet all the user-requirements?

Performance efficiency. Does the software have a fast response time? Are the inputs and outputs efficient? Does it efficiently access data?

Ease of use. Is it user-friendly? Are there sufficient resources to support the software? Does it have good error recognition/debugging capabilities?

Flexibility. Does the software provide various input and output options? Does it integrate well with other software?

Quality of documentation. Are there forums for reference such as "Frequently asked questions" (FAQs) or online tutorials available?

Manufacturer support. Are there downloadable product supports and updates available?

Kendall and Kendall [69] propose supplementary qualities of modelling that are important to consider in addition to the aforementioned guidelines when choosing a software suite. Three of these qualities include the *robustness* of the software, the software's *analysis and validation functionality*, as well as the software *licensing*. The software should be robust in the sense that the tool should be user-friendly and easy to interpret in order to simplify the development process. The software should have suitable analysis and validation functionality, such as built-in debugging capabilities in order to simplify the process of identifying errors throughout the DSS development and, finally, some software is licensed which may incur excessive costs, whereas others are free to use.

DSSs may soon become very complex systems during the development stages. They have to be logical in order to provide support to users during their decision making procedures. Therefore, in order to successfully develop a well-structured DSS, it would be beneficial to follow a *systems development methodology approach*. The purpose and varieties of this approach are discussed in the following section.

4.3 Systems development methodology approaches

Walters *et al.* [143] defines system development methodologies as a dependable collection of procedures to adopt, as well as tools and techniques to employ in order to make the development of a DSS successful. System development methodologies consist of a number of integral components that are also deemed relevant in the development of a DSS. These components comprise the *systems development life cycle*, as described by Kendall and Kendall [69] and is illustrated in Figure 4.4. First, problems, opportunities and objectives have to be identified, after which the information required by the user should be determined. Next, the system has to be analysed and, thereafter, the recommended system may be designed. Once the system is designed, it may be developed and documented in such a manner that it may be tested and maintained in the correct manner. Once the system is tested and cleared to be ready, it may be implemented and evaluated by the user. These phases serve as a basis for the construction of a DSS and may be customised according to the user's specific need for support. Three of the most popular system development methodologies are known as the *waterfall* methodology, the *agile* methodology and the *object-oriented* methodology:

Waterfall methodology. This methodology consists of a top down, exact approach where the complete process is documented sequentially as the phases are completed [38]. This methodology does not allow any backward changes (*i.e.* once a phase is completed, it cannot be returned to) and, thus only forward progression is allowed. Each phase requires the approval of the organisation or stakeholders before it can be marked as complete and before the next phase can commence. The advantages associated with this approach is

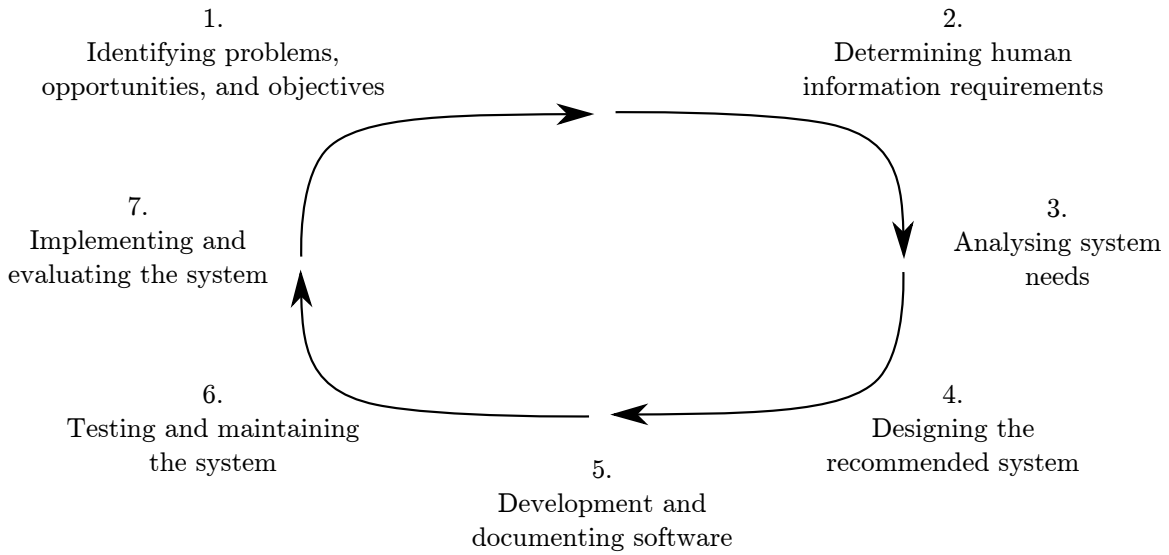


FIGURE 4.4: *The seven phases of the systems development life cycle for developing a DSS. Adopted from Kendall and Kendall [69].*

that it is a simple process to follow and each phase typically has set milestones that have to be completed in order for a next phase to commence. Furthermore, the information (*i.e.* documentation and requirements) associated with each phase is made known in advance. A major disadvantage associated with the waterfall methodology is that it makes the theoretical development of the DSS challenging since it does not include much revision time in each phase. Furthermore, the development process of such a system is quite time-consuming and is typically developed over long periods of time. Therefore, the risk is taken that the system might be accurately developed according to the specified requirements, but that the entire system is outdated by the time it is ready for use.

Agile methodology. This methodology makes use of iterative and incremental methods known as "sprints", which assists teams with coping with the unpredictability that exists when constructing a system. The original purpose of this methodology was to accelerate the system development process in order to overcome the disadvantages associated with other system development methodologies. This method also allows more focus towards successfully identifying and attending to user specified requirements. This method also attempts to develop a realistic DSS, instead of a DSS that works hypothetically (*i.e.* DSSs that are purely based on literature and that do not incorporate real-world instances). The advantages associated with the agile methodology is that it has a faster lead-time on delivering a functional DSS and that the developers may make modifications to any phase during any stage of the process [69]. Disadvantages associated with this methodology are typically due to flaws during the implementation phase (*i.e.* lack of realistic functional correlation to the DSS) or due to human errors (*i.e.* programming errors), rather than an incorrect methodological approach.

Object-oriented methodology. In contrast to the waterfall methodology, the object-orientated methodology consists of a bottom-up approach, which groups the DSS subsystems into objects or entities that contain data with similar characteristics (*i.e.* locations, people , events or other DSS components). This bottom-up approach allows for backward-changes and provides opportunities for revision, which is not possible in the waterfall methodology.

4.4 Testing, training and maintaining the system

Sufficient testing, training and maintenance of a system plays a vital role in the ultimate success of a system. The methods associated with this three-part quality measure is discussed in this section. In order to ensure that a system is tested effectively, other aspects such as quality assurance, verification, validation, implementation and maintenance have to be performed. These aspects are described in the sections to follow.

4.4.1 Quality assurance

According to Kendall and Kendall [69], quality assurance is defined as maintaining some pre-specified level of quality throughout the development stages of a product or service. This is important to ensure that all products or services conform to some standard that may be used to evaluate each individual product or service. Kendall and Kendall [69] propose three methods that may be employed to ensure quality when implementing software. The three methods are a *design* method, a *system documentation* method and a *testing and maintenance* method. The design method includes the design of systems according to a top-down or modular approach, where the top-down approach defines a broad overview of the system and each part of the system is refined and defined in more detail, while in a modular approach the system is divided into smaller parts that may be created independently and are usable in various systems. The system documentation method includes continuous documentation on software with relevant tools (*i.e.* procedure manuals that accurately document processes within a system) in order to accurately maintain the existing system. Furthermore, the testing and maintenance method includes continuous testing, maintenance and auditing of software with the help of experts that can ascertain the reliability of the system.

The use of the three methods may prove to be very useful, since problems may be detected early in the development stages and it is easier to resolve a problem during early stages than later when the problem has become more complex to solve. By making use of the method of top-down or modular approaches, it allows the developer to focus his/her attention on smaller parts of the system at a time, while keeping the bigger picture and goal in mind. The method of system documentation by Kendall has already been highlighted and discussed in §4.2.1 and the method of testing and maintenance will be discussed in the sections to follow.

McCall's factor model, presented by Galin [51], may also be used as a guideline to follow when wanting to ensure quality assurance of software. The model comprise eleven factors that may be divided into three groups, namely *product operation*, *product revision* and *product transition*. Product operation refers to the reliability and usability of the product and whether it operates correctly. Product revision refers to how easily the product can be maintained and whether it can easily be tested and product transition refers to the product's flexibility and whether it can be used in other contexts. The three groups along with their respective quality factors and descriptions are presented in Table 4.1.

The main goal of these three groups is to ensure that the software may be classified according to certain quality requirements, in order to inform the end-user of its capabilities. It has been noted by Galin [51], that many cases of end-user dissatisfaction results due to a lack of attention towards these basic requirements and, therefore the software suffers from poor performance in important areas such as maintenance, reliability, software re-use and training instances.

Group	Factor	Description
Product operation	Correctness	Free from any errors.
	Reliability	Performing consistently well.
	Efficiency	Performs in a well-organised manner.
	Integrity	Logical and consistent.
	Usability	Flexibility of use.
Product revision	Maintainability	Easy to repair.
	Flexibility	Easy to modify.
	Testability	Testable with accessible software.
Product transition	Portability	Transferable from one system to another.
	Reusability	Usablity in other existing systems.
	Interoperability	Working in conjunction with other systems.

TABLE 4.1: *McCall's factor model provides eleven factors divided into three groups that aims to achieve basic quality assurance requirements, as adapted from Galin [51].*

4.4.2 Testing the system

Sargent [116] defines *validation* as a method of confirmation that a model is developed to operate within a satisfactory range of accuracy when it is applied within the intended field of use. Thus, validation is simply the process of affirming that the correct system has been built according to the user's level of confidence in the system. On the other hand, *verification* is defined as the method of ensuring that the model structure and implementation is done correctly. Thus, verification is simply the process of affirming that a system has been built correctly and may be measured according to the user requirements that were presented and is continuously tested throughout the development of the model [116].

In most cases, however, it is very costly and tedious to determine model validity and, therefore tests and evaluations have to be conducted until the model displays a sufficient level of confidence in order for it to be considered valid [116]. In order for a system to be considered valid, it must exhibit some level of reasonableness, which may be measured according to five factors as provided by Bekker [10]. These factors include:

Continuity. If subtle adjustments are made to parameters there should exist similar subtle changes in the system output and results.

Consistency. Similar runs on the same system are required to produce similar outputs and results.

Degeneracy. The results provided by the system should reflect any changes made to the input parameters that were entered into the system.

Nonsensical conditions. If nonsensical input parameters are provided to a system, the system should be able to identify its absurdity and notify the user accordingly. Thus, rather than providing nonsensical results, the system should provide an error message.

Extremes. The system should be tested at its minimum and maximum extremes in order to ensure it acts correctly.

Validation is not intended to simply be a local process and it is, therefore, encouraged that these five factors must be tested continuously throughout the model's entire development process. Furthermore, Landry *et al.* [75] have proposed techniques that may be used in order to validate a model. These techniques are listed in Table 4.2.

Type	Description
Face validation	Acquire expert opinions from people that are knowledgeable in the field of interest, in order to gather their opinions on the accuracy of the model.
Tracing	Track specific entities throughout the model execution in order to observe their behaviour.
Internal validation	Perform a stochastic analysis on the variability that is present in the model.
Sensitivity analysis	Observe the effect that varying input parameters have on the output results of a model. This could also be done by comparing the model to other similar models.
Historical validation	Use historical data in order to determine whether the model behaves as it is expected to.
Predictive validation	Perform tests to validate whether the model's actual behaviour and expected behaviour correlate.
Events validation	Compare similarities in the events that occurred in the model to the distribution of events in the model.
Turing tests	Determine whether experts can distinguish between the actual system and the model results.
Spectral analysis	Determine whether the dynamic behaviour of the model is different from that of the system in the frequency domain.
Experimentation	Manipulate input variables in the real-world, as well as in the model, and compare outputs.
Convergent validation	Perform a comparison between the model's results and the predictions of experts.

TABLE 4.2: *Model validation techniques that are used on a frequent basis, as adopted from Landry [75].*

Even though software testing can become tedious, it plays a vital role in the success of any software since it ensures the quality of the end-system. Testing procedures verify whether the delivered product complies with the user-specifications. The user-specifications have to be comprehensive, complete and concise in order for testing to be effective. Testing procedures may be divided into two parts. The first part focuses on *non-functional requirement specifications* (*i.e.* does the system meet the user-specifications?), whereas the second part focuses on the *functional requirement specifications* (*i.e.* does the system do what it is required to do?). System developers are required to test systems on various levels in order to ensure that the system is robust on all levels. Some typical testing types are illustrated in Figure 4.5, as it is described by Inflectra [64].

Functional testing is typically divided into four parts, including *unit* testing, *integration* testing, *system* testing and *acceptance* testing. Functional testing is typically executed according to the design specifications as provided by the end-user. These testing methodologies are usually executed in the order that they are listed, and may be described as:

Unit testing. Unit testing tests each component or module of the software individually in order to ensure the software components work effectively on their own. This type of testing is typically performed in an automated manner.

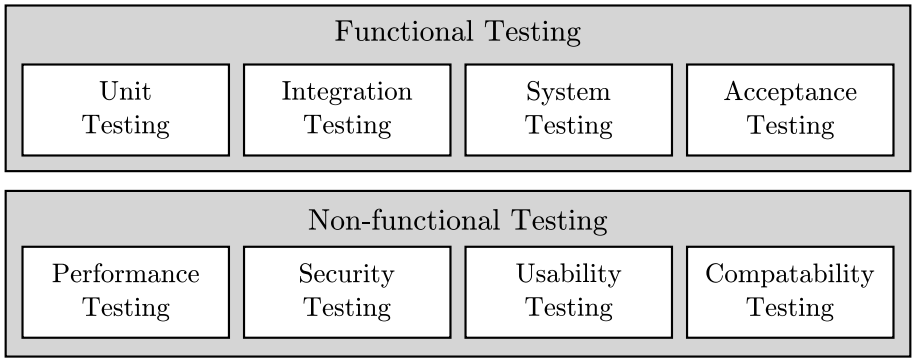


FIGURE 4.5: *Functional and non-functional testing methodologies, as described by Inflectra [64].*

Integration testing. Integration testing tests the various components or modules as a whole, by performing certain tasks. This type of testing is typically performed in both a manual and in an automated manner.

System testing. System testing tests the entire system for bugs and errors. The software is tested to observe whether it can execute certain tasks as required by the user, as well as testing the software on extreme data cases in order to see whether it still provides sensible results. This type of testing is typically performed in a data-driven manner, where test cases provide input parameters and various combinations of test data, in order to test both expected and random cases on the system.

Acceptance testing. Acceptance testing is the final part of functional testing and determines whether the software or system functions have met all of the predefined user-requirements.

Non-functional testing is also typically divided into four parts and includes *performance* testing, *security* testing, *usability* testing and *compatibility* testing. This type of testing measures the system according to some technical qualities, such as scalability, susceptibility and usability. These testing methodologies may be described as:

Performance, load and stress testing. Performance testing tests the load which a system can handle effectively, how it performs under different load variations and also whether the different load variations result in system failures due to too much stress incurred.

Security and vulnerability testing. Security testing tests whether software is confidential, whether it has integrity and whether it is authentic. Individual tests are also usually conducted to prevent unauthorised access to any of the software modules.

Usability testing. Usability testing tests whether the user had a good experience while using the software in terms of it's user-friendliness, performance, effectiveness and memorability.

Compatibility testing. Compatibility testing tests whether the product performs in a similar manner in different environments (*i.e.* web environment versus mobile application environment) and does not lose some of its functionality.

Furthermore, when functionality and non-functionality testing are executed, it is important to make use of real-time scenarios rather than simply performing ideal testing, since the actual product will be used by an end-user on various different real-world applications and not by a trained systems tester. System testing is critical and has to be performed effectively and thoroughly in order to prevent issues in a live environment. Testing should, therefore, be an

agile process that is conducted in every phase of system development and by every stakeholder involved. The different phases of software testing is illustrated in Figure 4.6 and discussed below.

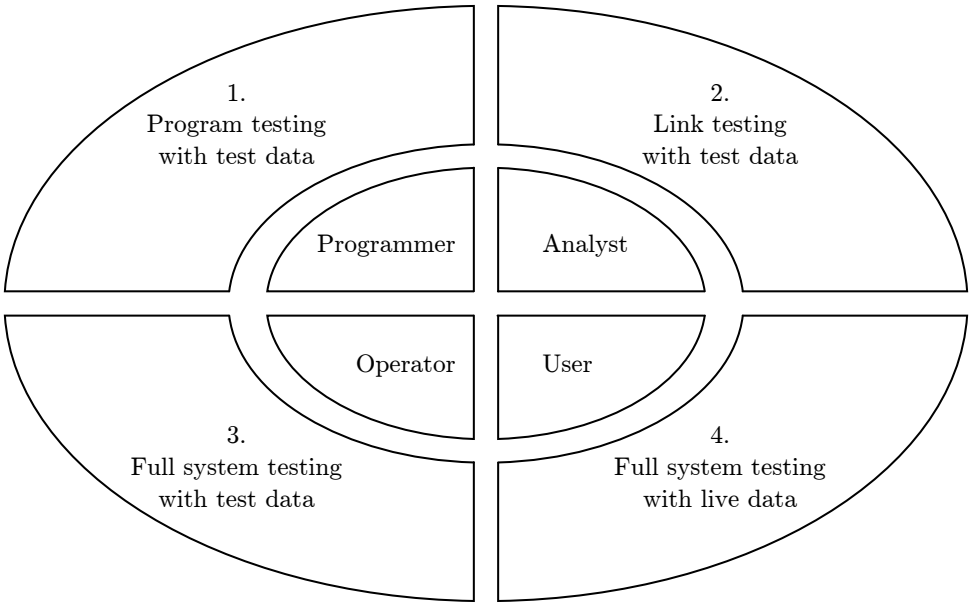


FIGURE 4.6: *Four phases for testing software and systems, as described by Kendall and Kendall [69].*

The first phase is testing the program with test data and the original *programmers* of the software are responsible for this. Furthermore, the systems *analyst* is also involved in this phase and acts as a coordinator of the testing process. The analyst observes the system output and advises the programmers on any problems which may arise. The analyst, therefore, has the responsibility of ensuring that programmers enforce the necessary testing techniques — they are in an advisory role and do not physically perform the task themselves. In this phase, testing techniques involve testing the system with both valid and invalid data, in order to check whether the system is able to run smoothly and identify errors. If errors are detected, the analyst reports this to the programmers, and the programmers have to reanalyse the process and fix the problem areas.

After performing the program testing, the system must undergo a *link testing procedure with test data*. This test checks whether interlinked programs work interdependently as was originally planned. In this phase, the *analyst* is responsible for creating test data for all the various scenarios that may arise in order to point out any underlying errors that may exist. If errors are detected, the analyst reports this to the *programmers*, and the programmers have to reanalyse the process and fix the problem areas.

After link tests have been concluded satisfactorily, the *full system* must be tested *with test data*. This test data is created as a collaborative effort by the system *operators*, as well as the *end-users*. In this phase, the system is tested with the objective of checking that the system's measure of error, ease of use, timeliness, and amount of down time are all adequate. This phase is also used to establish adequate procedure manuals that document the system and it's working.

Finally, after the entire system has been tested with test data, it is required to test the *full system with live data*. Live data refers to data that has been accurately processed and that has been confirmed by the analyst as "correct". The system is tested by using the live data in order to determine whether the system can return the same output. This phase in the testing procedure is, however, not always feasible, especially when new models with unknown output data are used in the model base component of Figure 4.2. In many cases, the full spectrum of

outputs may not be predictable from the start and, therefore, much of the live testing is done by the *user* when (s)he uses the software on real-world scenarios for the first time and reports possible issues back to the system *operators*.

4.4.3 Training the system

In order to ensure well-organised software roll-out with minimal end-user hesitancy and productivity losses, it is important to continuously keep deployment training in mind while developing software. Einstein famously said "Learning is experience. Everything else is just information." [55]. Training related to system functions is, therefore, important in order to translate how a system works under specified conditions. In many cases, it is very beneficial to have detailed training manuals available that a user can refer to when (s)he does not know how to operate the system. These training manuals are typically used to provide detailed information describing both the system interface and also to document how the system reacts or interprets certain commands.

Companies are expected to identify training needs according to the different skill levels of employees. In a system environment, the developers have to determine the skill level the end-users have and according to which degree they need to be able to interpret and understand the software they are using. In order for the end-user to understand the results or outputs provided by the system, they must be able to refer to some sort of manual that describes the various outputs and how to interpret them. This manual has to be maintained and updated on a regular basis, especially when software upgrades occur.

There are four important concepts to understand in order to effectively implement training, according to the Training Industry [33]. These four concepts include *modality*, *microlearning*, *marketing* and *maintaining*, and is referred to as the *Four M's*. Modality is the first that requires consideration, since different end-users require different learning approaches in order to effectively understand new software. Typical learning approaches exist in variations of visual approaches, auditory approaches, tactical approaches or experience approaches. By tailoring a training module to the end-user's needs, it enables them to learn to the best of their abilities. Microlearning is used to divide a training program into different, smaller activities and tasks that each have a specific learning outcome. This enables the end-user to interpret complex software in a less complex manner, while simultaneously providing learning reinforcement. By making use of microlearning, it also makes the training process less strenuous on the end-user, because they can learn little increments at a time according to the amount of knowledge they feel they can retain. Moreover, marketing is important during a new system or software launch in order to notify end-users of possible changes to software and how they can be trained to learn how to use these changes and, finally, proactive maintenance on training procedures for software has a large effect on the success of the system or software. This ensures that end-users can stay up to date with any changes made to software.

Employee training typically consists of different phases, including *orientation*, *training requirements*, *testing* and *evaluation* [35]. Orientation sessions or tutorials are important to provide an introduction to the system software that the user will be using. This gives the end-user the chance to familiarise himself/herself with the software and how to navigate it. Training requirements serve as a benchmark to ensure that all users have the same reference basis towards the software and that they have access to all the information and knowledge that would help them to utilise the software to the best of their abilities. Depending on the nature of the software, the training program or manual may be very basic or very extensive. The user can be tested on

their knowledge of the software, and an evaluation of their results during their test will provide valuable feedback on whether the training was sufficient, or whether the user requires additional training.

Another newer method used for training purposes is *simulation training*. Simulation training utilises computer software or other forms of basic equipment to realistically imitate real-world scenarios [137]. Simulation training teaches the end-user how to navigate certain tasks or how to perform certain actions in a variety of different scenarios, in order to ultimately prepare the user for similar scenarios that may occur in the real-world. For companies having sufficient resources, it is even more beneficial to incorporate simulation training as a part of their training programs since it provides an immersive learning environment. Research has shown that simulation based learning environments allow end-users to commit their experiences to memory as if the experiences were real [105]. This training approach does, however, have some disadvantages, such as that it has a relatively high price to implement, as well as that simulators are normal large pieces of hardware that contradicts the ever-increasing agile nature of global work forces. It is, therefore, more common that simulators are used in industries with end-users that require critical skill sets, such as pilots or military weapon assignment teams.

4.4.4 Implementing the system

The implementation phase is considered very important since it acts as a transition between the old system and the new system. In most cases, an existing system of some sort was in place before the new one was developed. Therefore, while the new system is being implemented, it is important that it is executed in such a way so as to make the end-users comfortable with it. Implementation of a newly developed system, typically requires training of some sort and is usually conducted by a systems analyst [69].

The reasons why new software or system implementations typically fall outside the boundaries of technology flaws. Vaughn [141] suggests considering the following factors when attempting to implement new software or systems in a company. The first factor is *the interaction between the technology and the company*, where the entire company should be able to relate to the change in systems or software. The end-users should feel a need for the new software in order to fully accept it, otherwise it could lead to conflict situations or rebellion against the new software or system. The second factor is *user participation*, where users should be given a sense of responsibility and control over the newly implemented software or system, in order to cultivate a positive attitude towards the system. The third factor is preventing *resistance*, by allowing end-users to express their ideas, fears and concerns regarding the new software or system in an accepting environment and also by providing representatives that are able to ensure that end-users understand the goals and purpose of the new system or software. The fourth factor is paying attention to possible *risks*. These risks could include underestimating the time and effort that is required in order to successfully implement a new software system, as well as effectively training the end-users. Another risk could be the assumption that the new system or software will work seamlessly from the initial implementation, whereas in reality it may need some minor adjustments to effectively perform tasks, or employees may require additional training in order to fully embrace the new system and its functionality. The fifth factor is *commitment*, where the end-user will form a sense of ownership if (s)he is involved in the implementation process from the beginning. The last factor is *planning*, which is a critical consideration in all new project implementations. The goals associated with the new software or system have to be clearly identified, and also how it is envisioned to achieve these goals. It is important to prepare an implementation time line and schedule in order to know when certain implementation phases have to be executed.

Evaluation also forms part of this phase, since users are able to provide feedback on the system and whether it sufficiently meets their needs in the environment it is used in. This may result in some cyclical work, as the analyst may have to fine-tune the system to comfortably fit into the user's environment and include some unforeseen additional features.

4.4.5 Maintaining the system

Kendall and Kendall [69] highlight two important reasons for performing maintenance. First, maintenance corrects and prevents software errors in the system and, second, it reinforces the system's ability to meet the organisational needs of the users. Maintenance is, therefore, also included as an ongoing process in the development and implementation of a system. Taute [127] describes maintenance as a life cycle which is continuously in motion. The maintenance life cycle is illustrated in Figure 4.7.

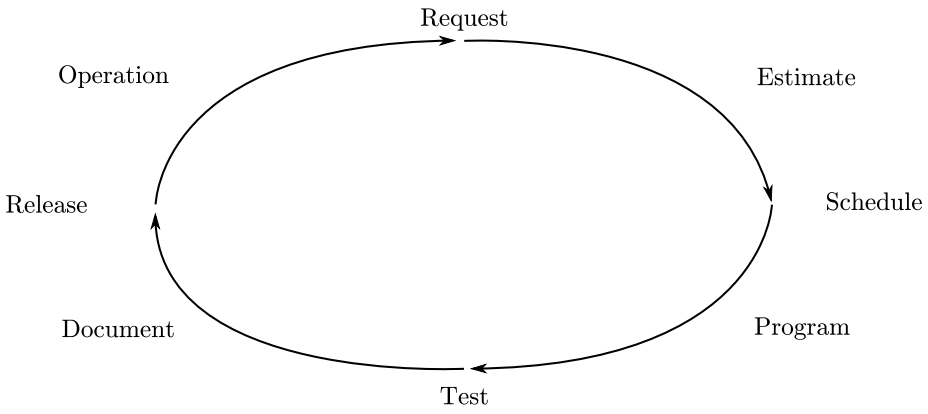


FIGURE 4.7: *The life cycle of maintenance, as adapted from Taute [127].*

The maintenance life cycle is initiated by a *request* for change on the current implemented system, however, the effort of performing these changes first has to be *estimated* and, thereafter, a *scheduled* release date is planned. This date refers to the date on which the change is expected to occur and should be implemented. Next, modifications are made to a controlled source code copy of the system in the *programming environment*, however, these changes first have to be *tested* and *documented* before they may be released. This testing and documentation is done according to the methods described in §4.4.2. During the test phase, the changes are verified and validated and all test runs are documented and analysed in order to observe the possible effects the new changes will have on the implemented system. If the testing and documentation phases are executed successfully, the changes may be formally implemented during the *release* phase. Once changes have been successfully implemented, they form part of the daily *operation* of the system. During the operation phase, end-users are actively utilising the new changes. They are able to use the system until another request is made for an additional change, in which case the entire life cycle is executed again in order to implement the additional change.

4.5 Chapter summary

The aim in this chapter was to present a comprehensive review of and provide a sufficient foundation for the understanding of decision support systems from in the literature. The chapter opens in §4.1 with a brief discussion on decision frameworks in general and continued to expand on decision support frameworks along with their various components in extensive detail in §4.2.

Section §4.2.1 focused on the general structure of a DSS and its components. Each component was described in detail along with the component's accompanying terminologies. The section also discussed the benefits that are associated with designing a good database, the five main types of database management systems, and guidelines for the design of a GUI. Section §4.2.2 focused on the selection of appropriate software that may be used for the development of a DSS and also provided guidelines that may be followed when developing a DSS, while section §4.3 discussed the system development life cycle that is relevant when developing a DSS. Sections §4.4.1–§4.4.3 discussed methods that may be used to ensure quality in developing a DSS and also suggested techniques that may be used to test, verify and validate the development of a DSS, as well as how to provide training to the end-users. Thereafter, the implementation and maintenance of a DSS was discussed in §4.4.4 and §4.4.5, respectively. The chapter finally closed in §4.5 with a brief review of the chapter contents.

Part II

Decision support system & simulation model

CHAPTER 5

Decision support system towards a simulation model

Contents

The material covered in this chapter provides a sufficient foundation for the understanding of the working of the DSS model that was developed in this study by constantly referring and consulting the research done in Chapters 2–4. The chapter begins by discussing the modelling software that is used to solve the problem, as well as why this specific modelling software was chosen, and is discussed in §5.1. Furthermore, in section §5.1.2 the various features associated with the modelling software are listed and their respective functions are discussed. The chapter then continues, by explaining the proposed DSS framework in §5.2.1, where the various components that form part of the DSS are explained in detail. Section §5.2.2 discuss the working of the two algorithms that are implemented in the DSS's model base component (*i.e.* the CVRP algorithm and the RCTVRP algorithm) by providing the pseudo code for each algorithm and explaining the working behind it. Furthermore, the chapter discusses the different object classes that form part of the developed model in §5.2.3. Object classes refer to the agents or objects in the model that are used to illustrate or perform certain tasks. In section §5.2.4, the working and visualisation of the graphical user interface is discussed. The GUI orchestrates various processes such as the model initialisation (which is discussed in §5.2.5), the optimisation of the algorithms in the AnyLogic modelling environment (which is discussed in §5.2.6) and, finally, the output results visualisation (which is discussed in §5.2.7). The chapter finally concludes with a brief chapter summary in §5.3.

5.1 Modelling software

Several researchers have acknowledged the difficulty associated with choosing a suitable simulation software from the various options that are available to users in modern times. The decision might be daunting to the developer since there are a substantial amount of considerations to include within the decision making process. Davis and Williams [36] have compiled a list of criteria that may be considered in order to aid the decision making process and ultimately choose a software environment most suitable for the problem that is to be solved. Important criteria to consider includes the *cost* of purchasing the software, the *comprehensiveness and flexibility* of the software, compatibility in terms of *integration* with other software systems, the availability of *training* courses and instruction manuals to understand the software, the *user-friendliness* associated with the software and, finally, the *hardware and installation* requirements in terms of memory and graphics cards required to process the system.

5.1.1 AnyLogic modelling software

The model developed in this thesis was developed in the ANYLOGIC 8 PERSONAL LEARNING EDITION 8.2.3 software suite [6]. This software package is flexible and comprehensive in the sense that it is a multimethod modelling technology that is capable of addressing complex business challenges with great adaptability and significantly high performance levels. It provides the opportunity to seamlessly deliver improved efficiency in business processes, as well as reducing risk in business environments. The software provides various platforms to analyse a problem and, subsequently provides insight into complex problems according to a level of detail that is acceptable by the user. These platforms include system dynamics, discrete event modelling, agent-based modelling and GIS animation features, which could all be beneficial in future work proposed to enhance the complexity of the current DSS and model. ANYLOGIC software may be integrated to work in conjunction with various other systems depending on the user requirements. The software is capable of capturing stochastic events that occur in real-life systems in a realistic manner and allows for real-time system design and analysis. The software also enables the user to design dynamic or conceptual models, depending on the level of abstraction that is required or envisioned by the end-user. The user-friendliness of the developing platform simplifies the process of building a model, due to its drag-and-drop functionality when adding new agents, functions, variables, parameters and other features. ANYLOGIC also has a built-in help manual and some tutorial videos on how to utilise certain features in the modelling environment. Furthermore, ANYLOGIC provides a sophisticated graphical modelling language which allows the user to visualise the model to a pleasingly realistic extent.

The fundamental programming language used in the ANYLOGIC software environment is JAVA code, which is currently regarded as one of the post popular programming languages utilised world-wide [113]. JAVA works well in the ANYLOGIC software environment due to the object-orientated nature of the language. Object-orientated programming allows attributes of data to interact with one another on various levels throughout the modelling environment. This allows the developer to design a hierarchical model architecture that interlinks agents and improves their connectivity by maintaining a flexible and sustainable structure that may be customised to a user's specifications as the need arises. All the attributes of object-orientated programming is synonymous with the intentions behind a simulation model and its working and, therefore serves as a good environment to implement the proposed DSS.

5.1.2 Software features

In the AnyLogic simulation environment, various features are provided that the developer may utilise in order to accurately represent agents and their interlinking relationships. The features aim to aid a user in effectively modelling the various agents in the system, as well as helping the user to understand the interactions between the agents that are present in the system. In order for the developed model to be a realistic representation of the real-world, the features that are used to model agents in the model have to be comprehensive and detailed. In order to accomplish this, the data required to describe agents can be stored, accessed and modified by the user. Agent behaviours and system dynamics are typically described by using a combination of functions, variables and parameters in order to accurately illustrate their various characteristics and individual functions. In some cases, the user may desire access to information regarding specific agents in the system, in order to obtain a more detailed analysis of the agent, or for debugging purposes.

Icon	Name	Description of use in simulation
	`Simulation Main`	Simulation Main acts as a platform for the user to configure the model before it is executed.
	`Agent`	Agents (also known as *object classes*) are the main building blocks of any AnyLogic model and posses qualities such as behaviour, memory, timing and contacts.
	`Function`	Functions return the value of an expression every time it is called in the model and may be used multiple times.
	`Variable`	Variables are often used to store results obtained through a model execution, or are utilised to model data units or object characteristics.
	`Parameter`	Parameters are often used to represent characteristics of a certain modelled object.
	`Collection`	A collection is used to represent a group of objects as a single unit and may be used to store, retrieve and manipulate aggregate data.
	`Excel File`	The Excel File acts as a connectivity tool that provides easy access to MS Excel files (*i.e.* .xls, .xlsx). These files may be read, updated and saved trough the connectivity tool.
	`Check box`	A check box acts as a control element that may be selected or deselected to perform a certain task.
	`Slider`	A slider is generally used to modify numeric variables or parameters within a specified interval.
	`Button`	A button is generally used to allow the user to influence the model interactively every time the button is clicked.

Table 5.1: *A summary of all the* AnyLogic *features used in the development of the model proposed in the thesis [56].*

Table 5.1 shows a list of features that where used in the development and design of the model presented in this thesis. A description, icon and name is provided for each feature. These features are used further on in this chapter to explain the model in more detail.

The life cycle of each agent in the simulation was modelled using a statechart. Each state marks a different stage in the agent's life where certain actions, events or a combination of functions, variables and parameters represent the agent's behavioural traits at that specific stage. In order for an agent to transition form its current state to another stage, a specific trigger must be activated and through repetition the agent ultimately moves through its entire life cycle. Triggers may differ depending on the circumstances surrounding the state. Table 5.2 explains the various types of triggers that may be used to activate a transition into a new state.

Icon	Trigger	Trigger description
	`Timeout`	A transition will occur after a predefined time interval, which may be stochastic or deterministic.
	`Rate`	Acts similar to a `timeout`, but in this case the time interval is determined by an exponential distribution and a parameter-defined rate.
	`Condition`	A transition will occur if a specified arbitrary boolean condition becomes true.
	`Message`	A transition will occur if a specific message is received from another agent.
	`Arrival`	The state is triggered if the agent of the state chart arrives at it's destination.

TABLE 5.2: *The triggers used in* ANYLOGIC *to activate transitions during the development of the model proposed in the thesis [56].*

5.2 Decision support framework

It is commonly known that the field of Operations Research makes use of decision support analysis by generating and testing different scenarios, in order to gain insight into, otherwise, speculative situations. The risk-constrained vehicle routing model proposed in this thesis is, therefore, designed with the purpose of investigating and gaining insight towards the risk associated with CIT vehicle routes by analysing data. A decision support framework is therefore developed and is described next. By using this framework, a model is developed in a modelling environment and the algorithms that are implemented in this section are described in §5.2.2. This model aims to mitigate risk along CIT vehicle routes.

5.2.1 Proposed DSS framework

The DSS framework proposed in this chapter, is designed based on the literature that was discussed in Chapter 4 and is illustrated in Figure 5.1. There are three main components that comprise the DSS, namely the database denoted by (C), the GUI denoted by (D) and the model base denoted by (E). These three components work in conjunction with one another to provide decision support to the user by analysing the user inputs denoted by (A) and proposing suitable vehicle routes that mitigate risk denoted by (F).

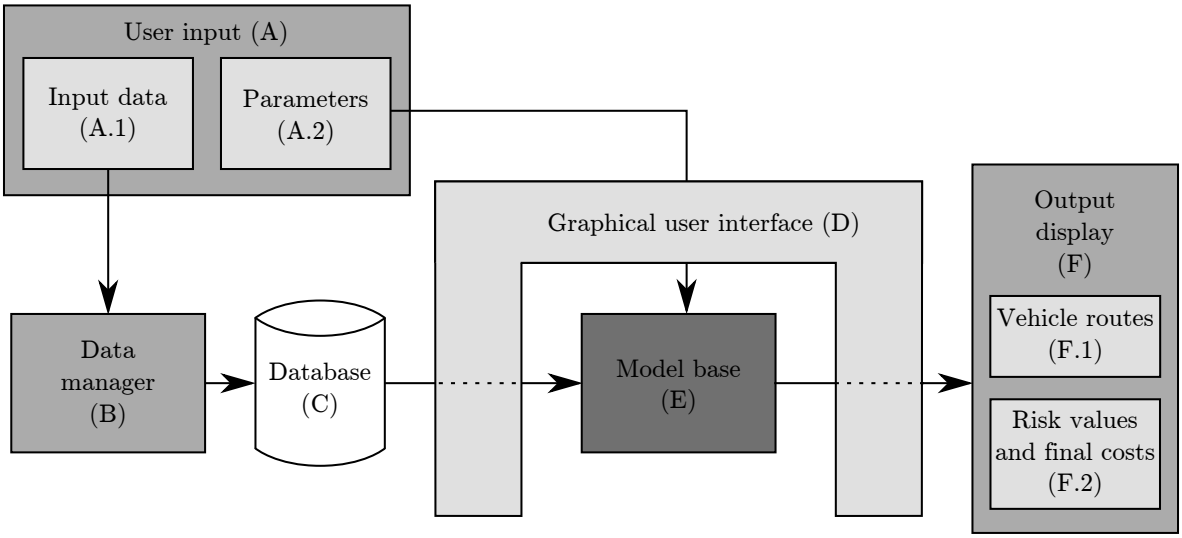


FIGURE 5.1: *The proposed DSS framework design for a RCTVRP model.*

The user plays a large part in the framework proposed in Figure 5.1 since a number of user inputs (A) are required in order for the other components in the DSS to function optimally. In order to provide a broad overview of the DSS, the user is required to provide input data, denoted by (A.1) in the figure, and then to manage and prepare this data, denoted by (B) in the figure. This user-provided input data is then relayed to the database (C). The user is also required to provide certain input parameters (denoted by (A.2) in the figure) to the model base, denoted by (E) in the figure. The user performs these tasks by working through the model's GUI (D). The model base requires the VRP input data (A.1) from the database (C), which includes data such as the number of customers, their locations and their respective demands. The model base also requires the user-defined parameters (A.2) that are defined through the GUI (D), which includes data such as vehicle capacity constraints, distance constraints and risk factor tolerances. Once the model base has received the inputs that it requires, the model in the model base may be executed. Finally, the model returns output results to the user through the GUI (D) in the form of an output display, denoted by (F) in the figure. This output display consists of visible vehicle routes, denoted by (F.1) in the figure, as well as the final risk values and costs associated with each route, denoted by (F.2) in the figure.

The input data (A.1) that is provided by the user must first undergo a data management process (B) to ensure that the data is in the correct format as required by the model and usable by the system. This data preparation process is illustrated in more detail in Figure 5.2 and consists of two main stages, namely the integration stage, denoted by (B.1), and the quality assessment and cleaning stage, denoted by (B.2). It is easy for human error to occur when importing data if a human is responsible for providing the data and, therefore, it is important to crosscheck the data and to ensure that it is in a usable format.

For the integration phase (B.1), the data is required to be in Microsoft's `Excel .csv` [92] format. When opening the spreadsheet, specific columns are allocated to specific data attributes relating to customer data, as well as specific `Excel sheets` allocated to certain attributes of the problem data. The most basic form of the required `Excel` spreadsheet layout is presented in Figure 5.3. The spreadsheet contains six sheets (which are indicated at the bottom of the figure) that support the working of the algorithm in the model base. The first sheet provides all *customer related information*, such as the number of customers, the x and y locations (coordinates) of the customers, as well as the demand of each customer. The remaining five sheets are reserved

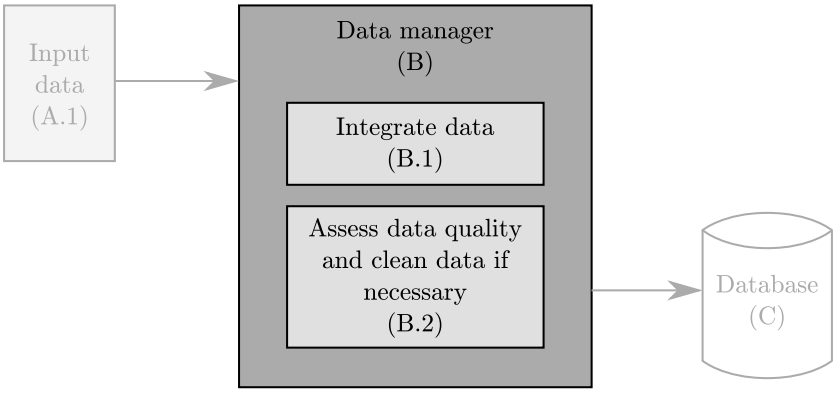


FIGURE 5.2: *Data preparation process involved in the DSS proposed in Figure 5.1.*

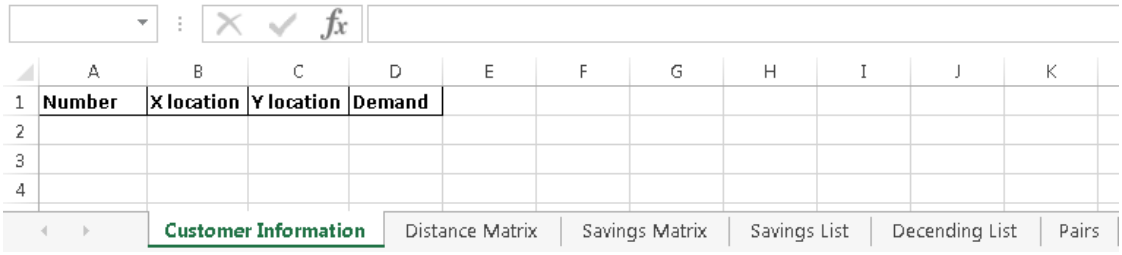


FIGURE 5.3: *`Excel .csv` file layout required by the model base.*

for model output. They are, therefore empty before the model is executed, but are filled with values as the algorithm is executed. The `Excel` sheets named *distance matrix*, *savings matrix*, *savings list*, *descending list* and the *pairs* are all integrate parts of the Clarke-Wright savings algorithm.

In more complex VRPs, other data such as time windows, risk factors and other parameters as discussed in Chapter 2 may also be required in the first sheet. All of these data entries (A.1) have to be integrated (B.1) into the required format of a Microsoft `Excel` spreadsheet in order for it to be read into the model base.

Furthermore, the data has to undergo a quality check and a cleaning process, denoted by (B.2) in Figure 5.2. This is illustrated in more detail in Figure 5.4. This secondary process is important in order to identify possible missing values in the data, denoted by (B.2.1), as well as to determine whether data values are realistic, correct and usable, denoted by (B.2.2), and also to identify any possible influential observations, denoted by (B.2.3). These influential observations could typically include data trends or identifying outlying and influential datum points.

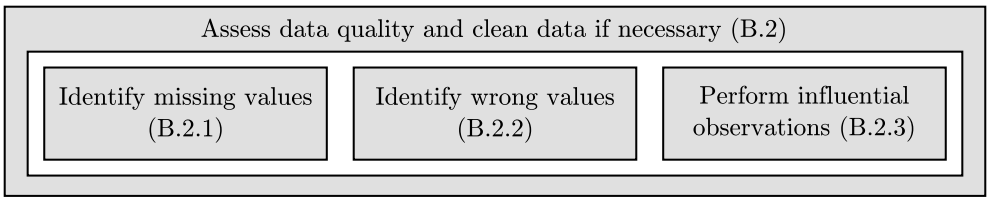


FIGURE 5.4: *Data cleaning process during data preparation in the DSS.*

It is commonly known that good quality input data provided to a model, results in a smooth execution of the algorithm used to solve the model which may provide better results [46]. It is, therefore, regarded very important to perform some form of data cleaning. Typical data cleaning

processes include removing *unwanted observations*, fixing *structural errors*, filtering *unwanted outliers*, and handling *missing data entries*. Unwanted observations refer to data entries that are regarded as duplicates or irrelevant data points, where duplicate points typically arise in the data collection process, and irrelevant points are points that have no notable contribution to the problem that is being solved. Structural errors refer to errors that typically occur during data transfer or data measurement, and is most commonly found in data with categorical features. Some common examples include typing errors and inconsistent use of capital letters (*i.e.* "X coordinate" versus "x coordinate") or mislabelled classes (*i.e.* "N/A" versus "Not Applicable"). Furthermore, unwanted outliers refer to data points that are drastically different than the rest in the set. Outlying data points are not a remarkable problem, however, the removal of such data points may help to improve the performance of a model, if there is a valid reason in removing these data points from the data set. Finally, when handling missing data points, it is commonly assumed that these data points may simply be left out, or be imputed based on the other data points in the set [46]. This is, however, not a fair assumption, since missing data points may provide important observations with respect to a data set and, therefore, it is more beneficial to indicate that certain data points are missing. This may be achieved by either labelling missing data as "missing" in the case where data is categorical, or by flagging and filling in the missing data for numerical data points. Once the data have been processed and cleaned, it may be imported into the database (C) after which it is relayed to the model base (E).

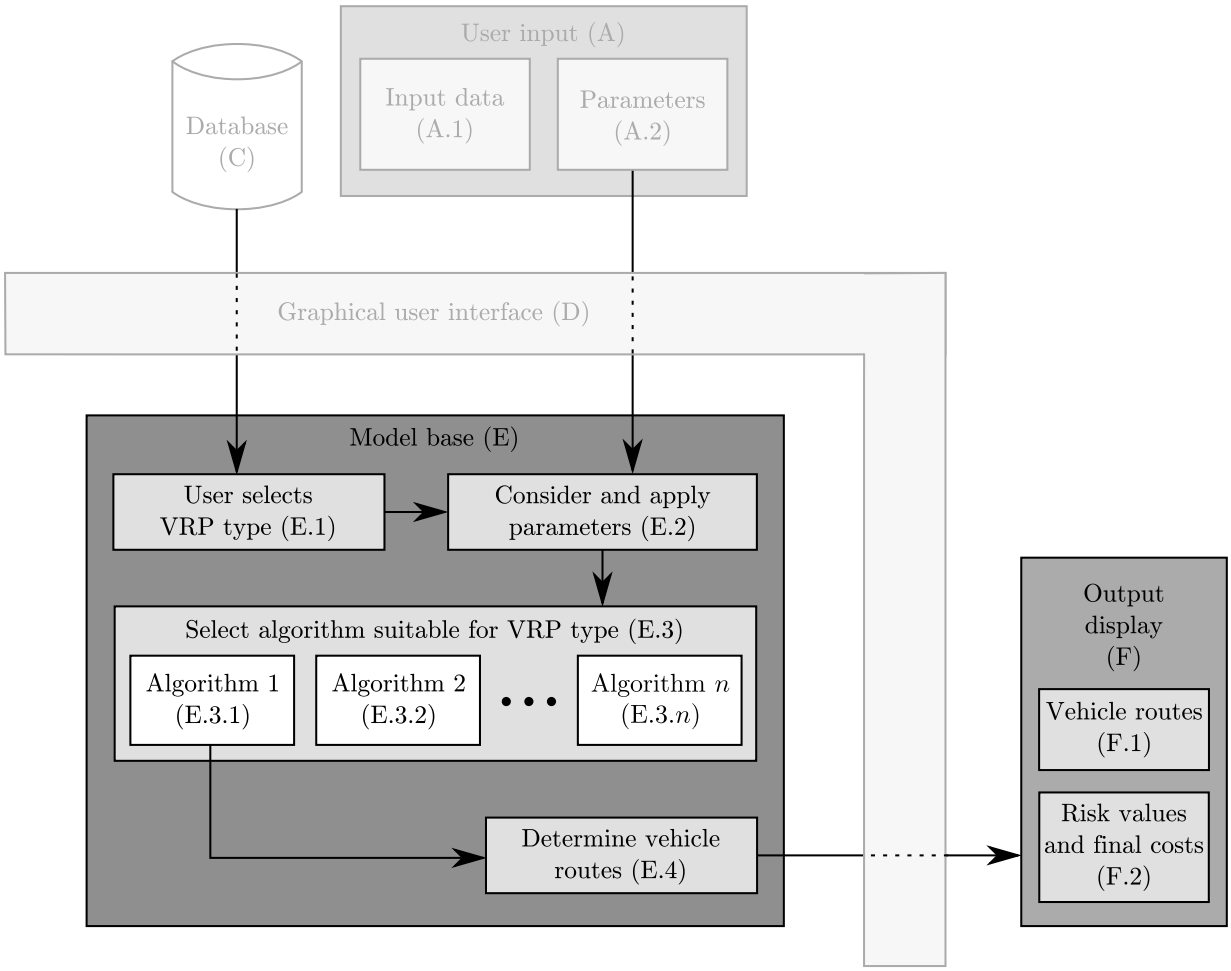


FIGURE 5.5: *The DSS model base for use in the proposed DSS framework in Figure 5.1.*

The model base (E) in the DSS framework in Figure 5.1 is based on the DSS framework from the literature that was presented in Figure 4.3 (and discussed in detail in §4.2). The reason for using Figure 4.3 as a basis to work from, is because the DSS presented in Figure 5.1 also makes use of different variations of model formulations depending on the type of VRP that is chosen or configured by the user. The detailed model base (E) for the DSS framework proposed in Figure 5.1 is illustrated in Figure 5.5.

The model base (E) in Figure 5.5 requires the VRP input data provided by the user and retrieves this information from the `Microsoft Excel` spreadsheet (as illustrated in Figure 5.3) that serves as the database (C) for this system. Furthermore, the model requires inputs from the user (A) in the form of parameters (A.2) in order to customise problems to the user's preference. These parameters are essential in order to determine the type of VRP that the user wants to execute (*i.e.* RCTVRP or CVRP). The GUI (D) allows the user to select the VRP type (E.1) by defining the parameters that are taken into account in the model base (E). The user is then allowed to either execute a normal CVRP with distance and capacity constraints, or the user may choose to incorporate risk mitigation. The model base then uses the parameters that were provided by the user and then, in turn, applies a suitable algorithm (denoted by (E.3) in Figure 5.5) to solve the model. Furthermore, the purpose of the developed model is to serve as a validation of the expected outcome, rather than a predict a certain outcome. The algorithms used in Figure 5.5 are based on variations of the Clarke-Wright algorithm that was explained in §2.2.1. The algorithm and its working is modified according to the constrains that were chosen by the user, and the chosen algorithm is used to solve the problem to propose vehicle routes. Finally, the vehicle routes returned by the algorithm and denoted by (E.4) in the figure is then relayed to the output display (F) which provides a visible output of the final routes (F.1) to the user, as well as an associated risk and impact estimation of the given routes (F.2) through the GUI (D) of Figure 5.5.

5.2.2 Algorithm frameworks

This section proposes the two algorithm frameworks that are used in the DSS model base of Figure 5.5 in component (E.3). The first is the infamous CVRP model (2.26)–(2.32) which was discussed in §2.1.2. The CVRP is used as the basis of the second algorithm framework, which is for the RCTVRP model (2.60)–(2.65). The RCTVRP was discussed in §2.1.6. Both of these frameworks are based on the working of the Clarke-Wright algorithm.

Algorithmic framework for the CVRP model

The CVRP is used to form the basic composition of the algorithm framework proposed in this thesis. The following assumptions are made in the CVRP framework

- that vehicles are constrained by some capacity limit,
- that all vehicles are homogeneous,
- that all vehicles start and end their routes at a common depot.

These assumptions are listed and discussed in §2.1.2, and are carried out and implemented in both frameworks discussed here. The working of the algorithm used to solve the CVRP model (2.26)–(2.32) is illustrated using the pseudo code in Algorithm **??** and is based on the working of the parallel savings approach illustrated in Algorithm 2.2. Algorithm **??** is executed for as long as the algorithm is still considering new pairs of linked customers in the savings list. The

algorithm has four main case considerations when linking customer pairs into existing routes. The first case is when customer 1, denoted as `W1` in the algorithm, and customer 2, denoted as `W2` in the algorithm, have *not been assigned* to routes as of yet. If the combination of `W1` and `W2` do not violate any constraints (*i.e.* distance or capacity constraints), they are linked together in a single route.

The second case is where `W2` has been assigned to a route and `W1` has not been assigned to a route. Then for every route present in the model, the algorithm has to consider whether the specific route can accommodate the addition of `W2` without violating any constraints. If `W2` may be added to the route, the algorithm must consider whether `W2` must be added at the start or at the end of the route. If `W1` is located at the start of the route, then `W2` is added at the start of the route, whereas if `W1` is located at the end of the route, then `W2` is added at the end of the route, in order to conserve the linked pair.

The third case considers the opposite of case 2, where `W1` has been assigned to a route and `W2` has not been assigned to a route. Then, for every route present in the model, the algorithm has to consider whether the specific route can accommodate the addition of `W1` without violating any constraints. If `W1` may be added to the route, the algorithm must consider whether `W1` must be added at the start of te route or at the end of the route. If `W2` is located at the start of the route, then `W1` is added at the start of the route, whereas if `W2` is located at the end of the route, then `W1` is added at the end of the route, in order to conserve the linked pair.

The fourth case considers when both `W1` and `W2` have been allocated to routes, but have not been allocated to the same route. It also considers whether the customers allocated to these routes are located at the end of these routes or somewhere in the middle of these routes. If at least one of the customers are located in the middle of these routes, the algorithm rejects the proposed location, because splitting up two routes in order to merge them is not allowed. If both the customers are, however, located at the ends of the routes, the two routes could possibly be merged according to the modification in §5.2.2.

If *W1* is allocated to route 1 and `W2` is allocated to route 2, then four possible scenarios with two possible outcomes each may occur. The first scenario is when `W1` is allocated "first" in route 1 and `W2` is allocated "first" in route 2. Route 2 may then simply be reversed and added at the start of route 1, or *vice versa*. The second scenario is when `W1` is allocated "first" in route 1 and `W2` is allocated "last" in route 2. Then route 2 may simply be added to the start of route 1, or route 1 may simply be added to the end of route 2. The third scenario is when `W1` is allocated "last" in route 1 and `W2` is allocated "first" in route 2. Then route 1 may simply be added at the start of route 2, or route 2 may simply be added to the end of route 1. Finally, the fourth scenario is when `W1` is allocated "last" in route 1 and `W2` is allocated "last" in route 2. Then route 2 may simply be reversed and added to the end of route 1, or *vice versa*.

The algorithm iterates through the customer pairs in the savings list until all customers have been allocated to valid routes and the costs of all the routes are minimised. Algorithm **??** is used in the model base of the DSS as a technique to optimise the problem by minimising the costs incurred along a route.

Algorithmic framework for the RCTVRP model

The RCTVRP model (2.60)–(2.65) was chosen for inclusion in this thesis in order to attempt to mitigate risk in CIT vehicle routes. The RCTVRP was first formulated by Talarico *et al.* [126] and bases it's working on the infamous Clarke-Wright algorithm that was discussed in §2.2.1. The original formulation of the Clarke-Wright algorithm had to be modified by Talarico *et*

al. [126] in order to accommodate the risk constraints of the RCTVRP model (2.60)–(2.65). Three modifications had to be made in the original CVRP formulation of model (2.26)–(2.32). These modifications are that:

1. the selection of the next pair of customers to add to a route, are from a restricted savings list (*i.e.* the list only includes the better savings pairs),
2. replacing the vehicle's capacity checking constraint with a risk checking constraint, although both the capacity constraint and risk constraint may be implemented at the same time as well,
3. checking the route risk in both directions of the newly formed route when customer pairs are linked into a route. The route orientation with the lowest incurred risk is used as the final route orientation.

The third modification had the largest impact on the Clarke-Wright savings algorithm since it has eight possible route-merging scenarios as a result. These scenarios are illustrated in Table 5.3 and are with reference to the RCTVRP model (2.60)–(2.65) described in §2.1.6. A capital letter R denotes a route that is reversed in direction, whereas a small letter r denotes a route in its original direction.

Position of customer i	Position of customer j	Merging possibilities
First	First	$R_2 \rightarrow r_1$ $R_1 \rightarrow r_2$
First	Last	$r_2 \rightarrow r_1$ $R_1 \rightarrow R_2$
Last	First	$R_2 \rightarrow R_1$ $r_1 \rightarrow r_2$
Last	Last	$r_2 \rightarrow R_1$ $r_1 \rightarrow R_2$

TABLE 5.3: *Possible route-merging scenarios that results when considering risk in a Clarke-Wright saving algorithm. Adopted from Talarico et al. [126].*

Before merging two routes into a single route in order to gain a saving s_{ij} (as described in §2.2.1), the algorithm first checks the positions of both customers i and j to check whether they are "first" or "last" in their respective routes. If the customer is "first" it means that it is the first customer the vehicle serves after leaving the depot, wheres if a customer is "last" it implies that it is the last customer served by the vehicle before the vehicle returns to the depot. Therefore, depending on the locations of customers i and j in their respective routes, there are eight possible route merging options as a result, as shown in Table 5.3. The modified Clarke-Wright algorithm has to check whether each of the possible route options are feasible (*i.e.* it does not violate a cost, a capacity or a risk constraint) and if one of the options is feasible, the second route is added to the first, or the first route is added to the second, depending on which merging option was chosen. An example of such a merge is shown in Figure 5.6. Consider the red route, denoted by r_1 and the blue route, denoted by r_2. Consider that the two round nodes indicated in red and in blue, respectively, have two be linked into a single route. Two possible merging scenarios exist. The first is the scenario where the red route remains in its original direction (r_1) and the blue route is reversed (R_2), which results in a "last-last" route merge ($r_1 \rightarrow R_2$). The second is the scenario where the blue route remains in its original direction (r_2)

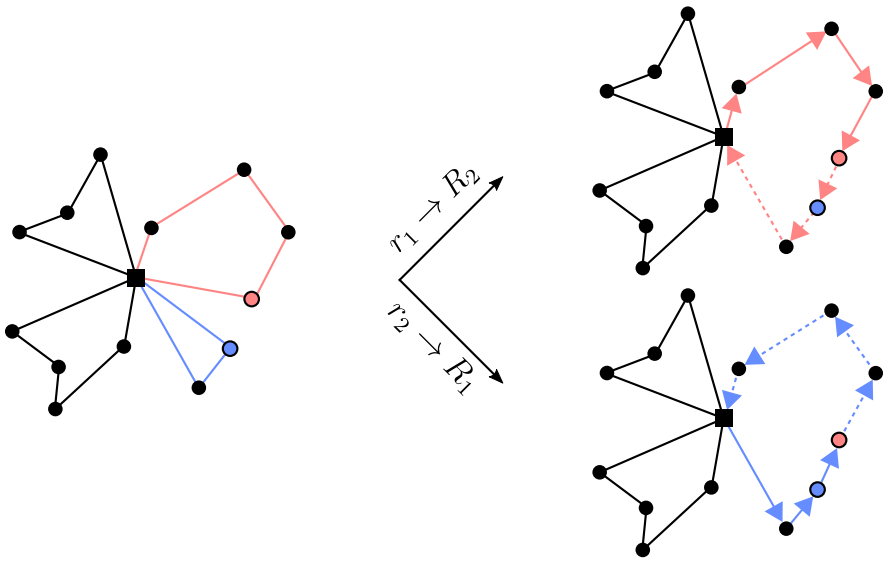


FIGURE 5.6: *Merging a route $r_1 \rightarrow R_2$ and $r_2 \rightarrow R_1$ for the RCTVRP model (2.60)–(2.65) by using the modified Clarke-Wright savings algorithm. Adopted from et al. [126].*

and the red route is reversed (R_1), which results in a "first-first" route merge ($r_2 \rightarrow R_1$). The algorithm will then check the feasibility of each of the resulting routes and if both routes are feasible, the route involving the lowest risk incurred is selected as the final route.

The basic working of the modified Clarke-Wright algorithm used to solve the RCTVRP model (2.26)–(2.32) is presented in the pseudo code in Algorithm 5.1. This algorithm was designed according to the modifications to the original Clarke-Wright algorithm that were discussed earlier in this section. In order to fully understand how the optimisation technique in the modelling environment works, the algorithm and its working must first be understood. The algorithm was developed according to a modular approach, as was discussed in the literature review in §4.4.1. The modular approach allows the developer to divide the development process into smaller, more manageable portions, and is also well suited for the ANYLOGIC modelling environment.

The algorithm is executed for as long as new pairs of linked customers in the savings list are still considered. The algorithm has four main case considerations when linking customer pairs into existing routes. The first case is when customer 1, denoted as `W1` in the algorithm, and customer 2, denoted as `W2` in the algorithm, have *not been assigned* to routes yet. If the combination of `W1` and `W2` do not violate any constraints, they are linked together in a single route. Once they are linked into a single route, the route orientation has to be determined according to third modification, which was listed earlier in this section. The risk is then calculated for each orientation of the route (*i.e.* in the reversed and original directions). This risk calculation is illustrated in Algorithm 5.2, where the algorithm calculates the possible risk incurred for the original route orientation, denoted by `Risk 1` and also the possible risk incurred for the reversed route orientation, denoted by `Risk 2`. If `Risk 1` is less than `Risk 2`, then `Risk 1` is selected as the final route orientation. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

The second case is where `W2` has been assigned to a route and `W1` has not been assigned to a route. Then for every route present in the model, the algorithm has to consider whether the specific route can accommodate the addition of `W2` without violating any constraints. If `W2` may be added to the route, the algorithm must consider whether `W2` must be added to the start or at the end of the route. If `W1` is located at the start of the route, then `W2` is added to the start

Algorithm 5.1: Solving the RCTVRP model using the modified CW algorithm.

```
1  for the length of the savings list do
2      if W1 and W 2 are not assigned to a route AND do not break any constraints then
3          CalculateRisk();
4      else
5          if W1 is assigned to a route AND W2 is not assigned to a route then
6              for every route present do
7                  if the route can accommodate W2 then
8                      if W1 is first in the route then
9                          add W2 at the start of the route;
10                         CalculateRisk();
11                     else
12                         if W1 is last in the route then
13                             add W2 at the end of the route;
14                             CalculateRisk();
15     else
16         if W2 is assigned to a route AND W1 is not assigned to a route then
17             for every route present do
18                 if the route can accommodate W1 then
19                     if W2 is first in the route then
20                         add W1 at the start of the route;
21                         CalculateRisk();
22                     else
23                         if W2 is last in the route then
24                             add W1 at the end of the route;
25                             CalculateRisk();
26     else
27         if Both W1 AND W2 are assigned to routes then
28             for Route 1 do
29                 for route 2 do
30                     if W1 and W2 are not assigned to the same route AND do not break constraints then
31                         if W1 is located first in route 1 AND W2 is located first in route 2 then
32                             Add route 2 to route 1;
33                             CalculateRisk();
34                         else
35                             if W1 is located first in route 1 AND W2 is located last in route 2 then
36                                 Add route 1 to route 2;
37                                 CalculateRisk();
38                         else
39                             if W1 is located last in route 1 AND W2 is located first in route 2 then
40                                 Add route 2 to route 1;
41                                 CalculateRisk();
42                         else
43                             if W1 is located last in route 1 AND W2 is located last in route 2 then
44                                 Add route 2 to route 1;
45                                 CalculateRisk();
46 end;
```

of the route, whereas if `W1` is located at the end of the route, then `W2` is added to the end of the route, in order to conserve the linked pair. The risk is calculated for each orientation of the route (*i.e.* in the reversed and original directions) according to Algorithm 5.2, and in a similar fashion as in case 1, if `Risk 1` is less than `Risk 2`, then `Risk 1` is chosen as the final route orientation. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

The third case considers the opposite of case 2, where `W1` has been assigned to a route and `W2` has not been assigned to a route. Then, for every route present in the model, the algorithm has to consider whether the specific route can accommodate the addition of `W1` without violating any constraints. If `W1` may be added to the route, the algorithm must consider whether `W1` must be added to the start of the route or at the end of the route. If `W2` is located at the start of the route, then `W1` is added to the start of the route, whereas if `W2` is located at the end of the route, then `W1` is added to the end of the route, in order to conserve the linked pair. The risk is calculated for each orientation of the route (*i.e.* in the reversed and original directions) according to Algorithm 5.2, and in a similar fashion as in the other cases, if `Risk 1` is less than `Risk 2`, then `Risk 1` is chosen as the final route orientation. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

Algorithm 5.2: Calculating risk in both route orientations.

```
calculate the risk of the original route orientation as Risk 1;
calculate the risk of the reversed route orientation as Risk 2;
if Risk 1 ≤ Risk 2 then
    Use the original route orientation;
else
    Use the reversed route orientation;
```

The fourth case considers when both `W1` and `W2` have been allocated to routes, but have not been allocated to the same route. It also considers whether the customers allocated to these routes are located at the end of these routes or somewhere in the middle of these routes. If at least one of the customers are located in the middle of these routes, the algorithm rejects the consideration, because splitting up two routes in order to merge them is not allowed. If both of the customers are, however, located at the end of the routes, the two routes may possibly be merged according to the modification discussed earlier in this section.

If W1 is allocated to route 1 and `W2` is allocated to route 2, then four possible scenarios with two possible outcomes each may occur, as was presented in Table 5.3. The first scenario is when `W1` is allocated "first" in route 1 and `W2` is allocated "first" in route 2. Then, route 2 may simply be reversed and added to the start of route 1, or *vice versa*. Once the two routes are merged, the route orientation must be determined, by calculating the risk for each orientation of the route (*i.e.* in the reversed and original directions) according to Algorithm 5.2, and in a similar fashion as in the other cases, if `Risk 1` is less than `Risk 2`, then `Risk 1` is selected as the final route orientation, and similar for the reversed case. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

The second scenario is when `W1` is allocated "first" in route 1 and `W2` is allocated "last" in route 2. Then route 2 may simply be added to the start of route 1, or route 1 may simply be added to the end of route 2. Once the two routes are merged, the route orientation must be determined, by calculating the risk for each orientation of the route (*i.e.* in the reversed and original directions) according to Algorithm 5.2, and in a similar fashion as in the other cases, if `Risk 1` is less than

`Risk 2`, then `Risk 1` is selected as the final route orientation, and similar for the reversed case. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

The third scenario is when `W1` is allocated "last" in route 1 and `W2` is allocated "first" in route 2. Then route 1 may simply be added to the start of route 2, or route 2 may simply be added to the end of route 1. Once the two routes are merged, the route orientation must be determined, by calculating the risk for each orientation of the route (*i.e.* in the reversed and original directions) according to Algorithm 5.2, and in a similar fashion as in the other cases, if `Risk 1` is less than `Risk 2`, then `Risk 1` is selected as the final route orientation, and similar for the reversed case. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

The fourth scenario is when `W1` is allocated "last" in route 1 and `W2` is allocated "last" in route 2. Then route 2 may simply be reversed and added to the end of route 1, or *vice versa*. Once the two routes are merged, the route orientation must be determined, by calculating the risk for each orientation of the route (*i.e.* in the reversed and original directions) according to the pseudo code illustrated in Algorithm 5.2, and in a similar fashion as in the other cases, if `Risk 1` is less than `Risk 2`, then `Risk 1` is selected as the final route orientation, and similar for the reversed case. A similar approach is followed for the reversed case (*i.e.* where `Risk 2` is less than `Risk 1`).

The algorithm iterates through the customer pairs in the savings list until all customers have been allocated to valid routes and the possible risk incurred is minimised on all routes. Algorithm 5.1 is used in the model base of the DSS as a technique to optimise the problem by minimising possible route risk incurred along a route. The implementation of the DSS in the ANYLOGIC environment is discussed in the sections to follow.

5.2.3 Object classes

The ANYLOGIC simulation environment provides the opportunity to develop models that are constructed by the use of independent components or agents, which are referred to as *object classes* in the ANYLOGIC context. Object classes allow flexible programming to the data structure of objects in die modelling environment, which also allows the developer to establish relationships between objects.

The complexity of an object varies according to its purpose in the modelling environment, and there exists three main attributes that describe the complexity of an object. The first is the *state* of an object, the second is the *behaviour* of an object and the third is the *identity* of an object. The state of an object may be described as a stage in the life cycle of an object that in turn describes the properties of an object, as well as the values associated with these properties. In the modelling environment, the state is typically expressed by means of variables and parameters that are established before model initialisation or during model execution. The behaviour of an object describes the functionality of an object, as well as how it will preform in certain states. In the modelling environment the behavioural characteristics of an object are typically expressed by means of object movement, state transitions and `Java` functions. Furthermore, the identity of on object describes object-specific attributes (*i.e.* attributes that are not dependant on the state of the object) that separate them from other objects in the environment. In the modelling environment, for example, some agents may exhibit similar behaviour and they might be in the same state, but they are still counted as individuals with their own identity.

In order to implement the algorithms described in §5.2.2 in the modelling environment, it is necessary to first determine which object classes are necessary in order to successfully represent the CVRP model (2.26)–(2.32) or the RCTVRP model (2.60)–(2.65). The primary agents required

to model these algorithms are the Simulation Main agent, the Main agent, the Depot agent, the Warehouse agent, the Vehicle agent, and the TrackingLine agent. Each of these are described briefly below and in more detail in the sections to follow.

One of the most important agents in the model is the Simulation Main agent. This agent is the first agent a user encounters when using the model. It provides the initial setup screen for the model, where the user is able to specify certain parameters and make the most of their decisions with respect to the construction of the problem. This agent is discussed in more detail in §5.2.4.

The most important agent is the Main agent, which serves as the primary modelling environment in which all the other agents reside. The Main agent is responsible for locating the Warehouse agents along with the Depot agent, as well as most of the functioning and behaviour related to the CVRP and RCTVRP model frameworks that ultimately calculate the number of Vehicle agents that are required. These functions are executed in a sequential manner. The Main agent is also responsible for housing the data sets that dynamically store relevant values related to the other agents, as well as providing relevant data that is used to provide sensible outputs to the user through the GUI. There are two main data sets, the first is the input data which comprises the Warehouse and Depot attributes and is denoted by Datasheet in the model. The second is the output data, which typically comprise the routes, costs and risk and is denoted by ResultsData in the model.

The Depot agent is simply located according to it's user specified coordinates, denoted by DepotXcoordinate and DepotYcoordinate in the model.

The Warehouse agent represents the network of customer nodes that require service by the vehicles. This agent stores customer attributes such as the name of the customer, denoted by Warehousename in the model, the customer demand, denoted by Demand in the model, and the location of the customer in terms of its coordinate location denoted as WarehouseXcoordinate and WarehouseYcoordinate in the model. Once the location of all the customers and the Depot are known, the euclidean distance matrix may be calculated. The distance from each customer to the common depot is saved as a variable in the Warehouse agent, denoted by DistanceToDepot in the model. Furthermore, the Warehouse agent has relationships to other agents such as whether or not the customer has been visited by a vehicle, denoted by a boolean variable Visited in the model, as well as if the customer was successfully added to a route during the execution of the Clarke-Wright savings algorithm, denoted by a boolean variable AssignedInSavingsRoute in the model.

The Vehicle agent represents the fleet of vehicles that are used to serve the customers. This agent describes attributes such as the current load carried by the vehicle, denoted by CurrentLoad, the current distance travelled by the vehicle, denoted by CurrentDist, as well as the current possible risk incurred by the vehicle, denoted by CurrentRisk in the model. Furthermore, the Vehicle agent contains a collection denoted as GenerateSavingsRoute which houses the customers that are assigned to the specific Vehicle agent. The Vehicle agent is also connected to the TrackingLine agent, which is a visual component used in the modelling environment that visually shows the routes travelled by the various vehicles. The TrackingLine feature is described in more detail in §5.2.4. Furthermore, in order to provide a visual interpretation of the routes the Vehicle agents are given movement functions that visually illustrate the route a vehicle takes and which customers it serves on the routes. The movement of a Vehicle may therefore be classified according to several states as shown in the state chart of the Vehicle in Figure 5.7. The possible states a vehicle can find itself in, include the *Locate* state, the *Move* state, the *Arrive* state, the *ReturntoDepot* state and the

TripComplete state. The transitions between the states are triggered by some of the triggers that were explained in Table 5.2.

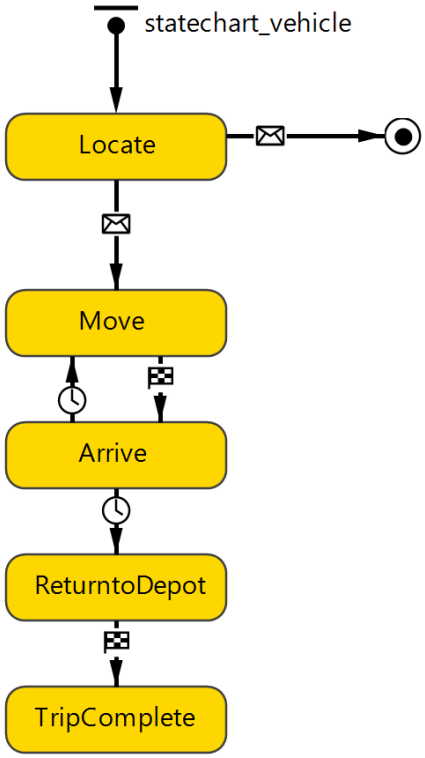


FIGURE 5.7: *The various states present in the life cycle of a vehicle agent as displayed in the* ANYLOGIC *modelling environment.*

First, a vehicle is *located* at the same coordinates as the common depot, since a vehicle always starts and ends its route at the common depot. The vehicle then receives a `message` trigger that prompts the vehicle to start its route and the vehicle will then transition to the *move* state. The vehicle will then move to the next customer allocated to its route and once it `arrives` at the coordinates of that customer, it will transition to the *arrive state*. Once a vehicle is in the arrive state it will send a signal to the customer it is at and that customer will change its colour from yellow (*i.e.* "not visited") to green (*i.e.* "visited"). Once the vehicle receives some `timeout` trigger (*i.e.* after the time that it takes to service the customer is finished), the vehicle will return to the *move* state. This iterative transition between the *move* and *arrive* states will continue until the vehicle serves the last customer that is allocated to its route. If the vehicle is in the *arrive* state and it is the last customer allocated to the route, the vehicle will receive another `timeout` trigger after its service time is finished and the vehicle will *return to the depot*. Finally, once the vehicle `arrives` back at the depot, the vehicle route is completed and the vehicle will enter the final *TripComplete* state.

5.2.4 Graphical user interface

There are two GUIs that the user encounters in this model. The first GUI the user encounters is the GUI presented by the `Simulation Main` agent and it forms part of the model initialisation which is discussed in more detail in the next subsection. This GUI allows the user to interact with the model by making use of visual icons, indicators and instructions. The visual representation and prompts provide a user-friendly platform for the user to define the problem and also allows

the user to customise the problem to his/her preference. A screen shot of the GUI as displayed in AnyLogic is presented in Figure 5.8.

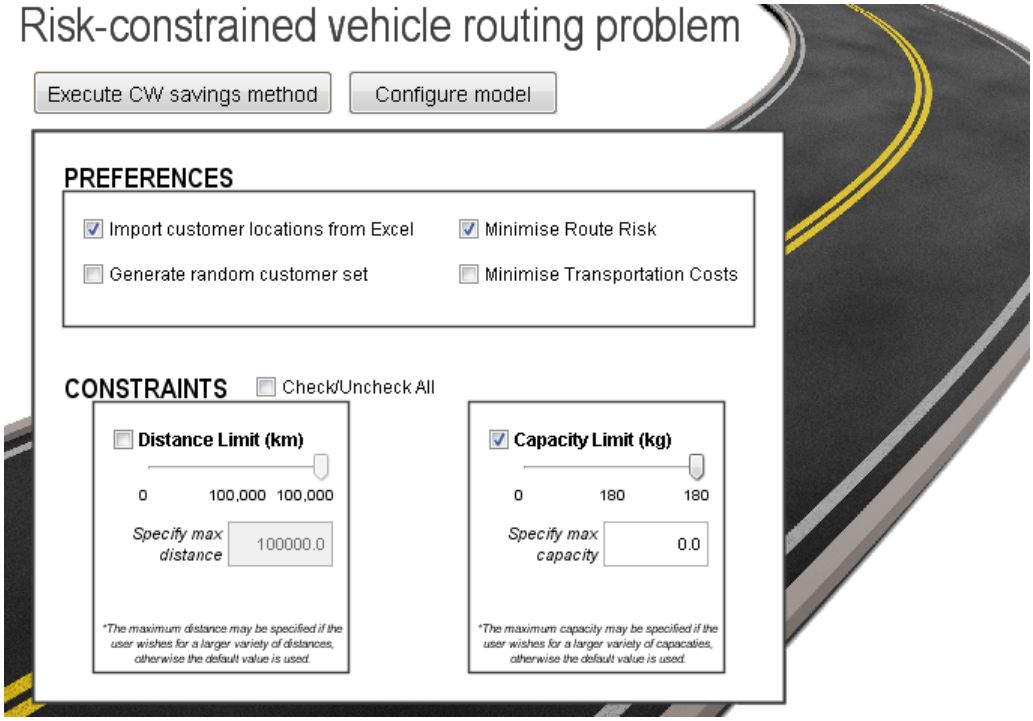


Figure 5.8: *The initial GUI in AnyLogic that the user encounters in the model.*

The second GUI the user encounters is the GUI presented by the `Main` agent and it forms part of the model visualisation which is discussed in more detail later in this section. This GUI provides a visual representation of the agents in the model and allows the user to visually interpret the problem by viewing and analysing the vehicle routes in a more practical manner.

5.2.5 Initialisation of model

In order to setup the model according to user preferences, the user is prompted to configure the model through the GUI presented by using the `Simulation Main` agent and which is illustrated in Figure 5.8. In §5.2.1 it was mentioned that the DSS requires data to be read in from an Excel file. The model, therefore provides the user with two checkbox options that allows the user to either `Import customers locations from Excel` or to `Generate random customer set`. The user will typically only generate a random customer set for illustrative purposes and will generally import customer data from Excel. Should the user select the `Generate random customer set` checkbox, the model will generate feasible random `WarehouseXcoordinate` and `WarehouseYcoordinate` coordinates for each `Customer` and also assign it a random `Demand` value. Should the user choose to import customer data from Excel, the `PlaceWarehouses` function is executed, which retrieves the customer data from the `Datasheet` and adds all the customers in the network to a collection of customers denoted as `collection_warehouses`. The function locates each `Customer` at its `WarehouseXcoordinate` and `WarehouseYcoordinate` coordinate location, and assigns each `Customer` a respective `Demand` value and adds all the demands to a collection named `collection_demand` with entries that correspond to that of the `collection_warehouses` collection. Furthermore, the `DistToDepot` function calculates the Euclidean distance from

each Cutsomer agent in collection_warehouses to the common Depot agent and saves each entry in a collection named collection_DistToDepot. In ANYLOGIC, there exists an inherent function that may be used to calculate the distance between two nodes (*i.e.* between two customers' x-y coordinates) and works according to the Euclidean distance function calculated as

$$d = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}, \tag{5.1}$$

where d denotes the distance between node 1 and node 2. Furthermore, the user may choose to execute one of two objectives for the VRP *i.e.* either to Minimise route risk, or to Minimise transportation costs, by selecting the appropriate checkbox. Should the user select the Minimise transportation costs checkbox, the model will automatically execute the CVRP model (2.26)–(2.32) by using Algorithm **??** and should the user select the Minimise route risk checkbox, the model will automatically execute the RCTVRP model (2.60)–(2.65) by using Algorithm 5.1. The user is able to *specify a maximum distance* in a textbox in order to adjust the maximum range of the Slider. The user may then adjust the distance Slider to adopt a MaxDistanceLimit that is enforced on the vehicle routes. In a similar fashion, the user is able to *specify a maximum capacity* in a textbox in order to adjust the maximum range of the capacity Slider. The user may then adjust the Slider to adopt a MaxCapacityLimit for a vehicle that is enforced on the vehicle routes.

If the user chose to minimise route risk by selecting Minimise route risk, the minimum allowable risk threshold that may be enforced in the problem must first be determined in order to ensure that the solution is valid — the method for calculating the minimum allowable risk threshold was discussed in §2.1.6. The CalculateRiskThreshold function is executed when the "Configure model" button is selected and calculates the minimum allowable risk threshold by establishing the possible risk of each route if a single vehicle were to service a single customer (*i.e.* the number of vehicles are equal to the amount of customers). Therefore, a collection named InitialRisk is made by multiplying the corresponding entries of collections collection_demand and collection_DistToDepot. The CalculateRiskThreshold function then checks which entry in the InitialRisk collection has the highest value and sets a variable named MaxRiskThreshold equal to that value. The minimum allowable risk threshold is, therefore equal to MaxRiskThreshold.

5.2.6 Optimisation with the RCTVRP algorithm

After initialisation, the user may select the "Run" button. The "Run" button executes the algorithm that was selected by the user which is either the SaveTestCVRP function for the CVRP model (2.26)–(2.32) or the SaveTestRCTVRP function for the RCTVRP model (2.60)–(2.65), depending on the checkbox which was selected by the user. This action also translates the user input Variables provided through the GUI in the Simulation Main agent to model input Parameters for the model in the Main agent. The equivalent model input Parameters for the equivalent user-provided input variables Variables are shown in Table 5.4 and will henceforth be referred to in their model input parameter equivalents.

Next, a number of functions are performed to solve the model. These functions include the working of the Clarke-Wright savings algorithm and its method of optimal route development. This section of the model is the most complex since it incorporates many functions, various

User input	Model input
CapacityLimit	VehicleCapacity
DistanceLimit	DistanceConstraint
ImportCustomers	ImportExcelLocations
GenerateRandomCustomers	GenerateRandomCustomerSet
MinimiseRisk	MinimiseRiskCosts
MinimiseTransportCosts	MinimiseTravelCosts
RiskThreshold	RiskThresholdUsed
MinRiskThreshold	MinimumRiskThreshold

TABLE 5.4: *Model input parameter names for the corresponding user input variable names from the GUI.*

variable constructs and all of the intricate model details in order to effectively solve the problem and provide a sufficient solution.

In order to implement the algorithms discussed in Algorithm **??** and Algorithm 5.1, they require slight adaptions in order to function effectively in the ANYLOGIC modelling environment. In the remainder of this section, these adaptions along with other more intricate details of the optimisation function is discussed in order to provide an overview of the process followed in order to create vehicle routes, by making use of the software features as described in §5.1.2. The functions for the CVRP model (2.26)–(2.32) and RCTVRP model (2.60)–(2.65) are discussed in parallel since the RCTVRP model (2.60)–(2.65) builds on the CVRP model (2.26)–(2.32) — a clear distinction is provided where the RCTVRP function has additional features that are not included in the CVRP.

The entire CVRP algorithm is located in the SaveTestCVRP function and, similarly the entire RCTVRP algorithm is located in the SaveTestRCTVRP function. Upon execution of either of the two algorithms the function starts by establishing the number of nodes present in the model by accessing the Datasheet and determining the number of rows (that represent the customers) plus the addition of the common depot and subsequently stores the number of customers in a variable, denoted as NumberWarehouses. Furthermore, the model proceeds to establish the number of vehicles present in the model by determining the size of the collection of vehicles, denoted as collection_vehicles, by using an inherent ANYLOGIC function named size() — the number of vehicles will initially be zero.

Before the algorithm may be executed, certain features of the Clarke-Wright algorithm have to be established first. The first of these is to establish the distance matrix containing the distances travelled between each pair of customers in the network including the common depot. The pseudo code for establishing the distance matrix is given in pseudo code form as Algorithm 5.3.

Next, it is required to allocate sufficient space for the distance matrix by declaring a variable of the type Cost = [NumberWarehouses][NumberWarehouses] which represents the matrix — the matrix is allocated enough space to accommodate all of the customer nodes in the network, as well as the common depot. It is assumed that the depot is located at node 0 and that the customers are located at nodes $1, \ldots, N$ as discussed in §2.1.2. The algorithm, therefore considers all the nodes in order to determine the distance between them. If node $i = 0$ (depot) and node $j = 0$ (depot), the distance (Cost[i][j]) between node i and j is equal to zero, because it is the same node. If node $i = 0$ (depot) and node $j \geq 0$ (some customer), the distance

Algorithm 5.3: Establishing the distance matrix for the set of nodes in the network.

```
1  Create space for the distance matrix, with space equal to the number of nodes in the
   network;
2  for each node i in the network do
3      for each node j in the network do
4          if node i = 0 and node j = 0 then
5              the distance is zero;
6          else
7              if node i = 0 and node j ≥ 0 then
8                  the distance is equal to the distance between node j and the common
                   depot;
9          else
10             if node i ≥ 0 and node j = 0 then
11                 the distance is equal to the distance between node i and the common depot;
12         else
13             the distance is equal to the distance between node i and node j;
```

(V `Cost[i][j]`) between node i and j is equal to the distance between the depot and customer j. If node $j = 0$ (depot) and node $i \geq 0$ (some customer), the distance (V `Cost[i][j]`) between node i and j is equal to the distance between the depot and customer i and, if node $i \geq 0$ (some customer) and node $j \geq 0$ (some customer), the distance (V `Cost[i][j]`) between node i and j is equal to the distance between the two customers.

The savings matrix containing the savings achieved when linking any customer pair into a single route (the savings matrix does not include the depot) must also be established. Establishing the savings matrix is given in pseudo code form as in Algorithm 5.4.

It is also necessary to allocate sufficient space for the savings matrix by declaring a variable of the type V `Savings = [`V `NumberWarehouses-1][`V `NumberWarehouses-1]` — the matrix is allocated enough space to accommodate all of the customer savings that can be achieved. The algorithm, therefore considers all the customers in order to determine the savings achieved by linking them into a single route. If node i and node j are the same node, then the savings (V `Savings`) is zero and if node i and node j are not the same node, then the savings is calculated according to the savings formulation in equation (2.78). Hereafter, the savings matrix is required to be in the form of a savings list of the type V `SavingsList = [`V `ListLength]`, where V `ListLength` denotes the number of unique customer pairs in the savings matrix. The entries in the matrix are added to the savings list by going through each savings entry in the savings matrix and adding it to the list without adding the same customer pair twice. A `counter` is introduced in order to ensure all individual entries from the V `Savings` Matrix are carried over to the V `SavingsList`.

After establishing the savings matrix, the various savings have to be sorted from highest to lowest and this is achieved by using a Bubble Sort Algorithm[1] which is given in pseudo code form as Algorithm 5.5.

Since the Clarke-Wright savings algorithm sorts the savings from highest to lowest (as discussed

[1] A bubble sort algorithm is a simple version of a sorting algorithm that continuously runs through a list to be sorted by comparing each pair of adjacent entries and switches them if they are not in the correct order.

Algorithm 5.4: Establishing the savings matrix for the set of customers in the network.

```
1  Create space for the savings matrix, with space equal to the number of customers in the
   network;
2  for each customer i in the network do
3      for each customer j in the network do
4          if node i is equal to node j then
5              the savings is zero;
6          else
7              the savings is calculated according to equation (2.76);
8  Create space for the savings list with space equal to the number of customers in the
   network;
9  Initialise a counter with the length of the savings list;
10 Establish a counter counter=0;
11 for  each customer i in the network do
12     for each customer j in the network do
13         the list entry equal to the number of the counter is equal to the savings between
           customers i and j;
14         increment counter;
15     increment j;
```

Algorithm 5.5: The bubble sorting algorithm.

```
1  Establish a holder variable as holder;
2  for the length of the savings list do
3      if the current entry value (i) is less than the next entry value (i + 1) then
4          holder = current entry value (i);
5          the current entry is given the value of the next entry (i = i + 1);
6          the next entry is given the value of the holder (i + 1 = holder);
```

in §2.2.1), the bubble sorting algorithm works through the list and switches entries until the entries with the highest values are at the top of the list. The algorithm therefore runs through the number of the customers in order to establish whether or not they should be switched. If the current entry value, denoted by `SavingsList[`i`]`, is less than that of the next entry, denoted by `SavingsList[`$i+1$`]`, the algorithm saves entry i in a place holder, denoted by `holder = SavingsList[`i`]`. The algorithm then allocates the value of entry i+1 to entry i by overwriting it *i.e.* `SavingsList[`i`] = SavingsList[`i+1`]`, and then allocates the holder's value to entry $i+1$ (*i.e.* `SavingsList[`$i+1$`] = holder`). Therefore, entry `SavingsList[`$i+1$`]` is now ranked higher than entry `SavingsList[`i`]`.

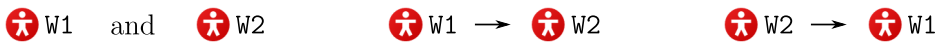
Furthermore, the customer pair associated with each savings value entry in the savings list are also arranged in a list format similar and corresponds to that of the savings list and is denoted by `Pairs[][]`. Other features that describe the customer pair are also arranged in similar list formats which correspond to that of the savings list, such as `PairDistance[]` and `PairCapacity[]`, where `PairDistance[]` refers to the distance between a customer pair and where `PairCapacity[]` refers to the combined capacity of a customer pair. For example if a customer pair consists of customer k and customer l and if the customer pair is the tenth

entry in the pairs list, the pair is denoted as Pairs[k][l], the pair distance entry is denoted as PairDistance[10] and the pair capacity entry is denoted as PairCapacity[10].

After the savings list along with its accompanying customer pairs list have been constructed the Clark-Wright algorithm may commence. The algorithm runs through the savings list, starting at the top with the highest possible savings values and works its way down to the bottom of the list (*i.e.* it considers all entries in the list, each entry denoted by $q = 1, \ldots, Q$, where Q is the total number of entries). For each savings entry, the algorithm determines whether the customers in the pair have or have not been assigned to a vehicle route, as well as whether or not the addition of the customer pair will violate any of the constraints in the problem. If the considered pair is valid, it may be added to the route.

The algorithm initialises two customer agents W1 and W2, where customer W1 is the first customer in the customer pair and customer W2 is the second customer in the customer pair under consideration. Furthermore, a variable denoted by TripDistance is set equal to the PairDistance[q] of the customer pair under consideration for the iteration. There are four main possible cases that could occur when considering a customer pair and whether or not it should be included in the route.

In the first case, neither of the two customers under consideration have been included in or assigned to a vehicle route (as illustrated in Figure 5.9(a)). In this case the boolean internal variable denoted as AssignedInSavingsRoute which is associated with the customer agents (*i.e.* customer W1 and customer W2) is equal to zero and, therefore indicates that neither of the customers have been assigned to a vehicle route. The algorithm continues by assessing whether the customer pair's combined demand (PairCapacity[q]) is less than or equal to the user specified vehicle capacity limit VehicleCapacity and that the TripDistance along with the distances travelled to the depot is less than or equal to the user specified distance constraint DistanceConstraint. If these constraints are not violated, the customer pair may be added together in a new route in the case of the CVRP model (2.26)–(2.32), as is ilustrated in Figure 5.9(b). In the case of the RCTVRP model (2.60)–(2.65) the risk constraint also requires consideration. In the case of the RCTVRP model (2.60)–(2.65), however, the algorithm defines two place holder variables for the risk, denoted as RiskSaver1A and RiskSaver1B. The risk is calculated according to equation (2.59) and since the risk has to be considered for both orientations of the route (as discussed in §5.2.2) the algorithm calculates two possible risk outcomes for the route.



(a) *Unassigned customers* (b) *Original orientation.* (c) *Reversed orientation.*

FIGURE 5.9: *Two possible route orientation outcomes for the first case in the algorithm. In (a) customer W1 and customer W2 are illustrated as unassigned customers, in (b) the original orientation of the route (i.e W1 → W2) is illustrated and in (c) the reversed orientation of the route (i.e W2 → W1) is illustrated.*

The first orientation is the original orientation *i.e.* where customer W1 is served first and customer W2 is served second and the second orientation is the reverse orientation *i.e.* where customer W2 is served first and customer W1 is served second. The possible risk for each of these orientations are calculated and stored in the risk placeholders named RiskSaver1A and RiskSaver1B, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk is also lower than the user defined risk threshold, denoted as MinimumRiskThreshold.

If RiskSaver1A achieves the lowest possible risk for the customer pair and is also lower than the MinimumRiskThreshold then the original route orientation is used, as illustrated in Figure 5.9(b). If, however, RiskSaver1B achieves the lowest possible risk for the customer pair and is also lower than the MinimumRiskThreshold then the reversed route orientation is used, as illustrated in Figure 5.9(c). Once it is established that none of the constraints (*i.e.* capacity, distance or risk constraints) are violated, a Vehicle agent is assigned to serve the customers. In order to acknowledge the assignment of both of the customers to a route, both of the customer agents' internal variables are adapted so that the value of the boolean variable AssignedInSavingsRoute is set equal to one and that the vehicle agents' internal variables are adapted so that the vehicle's CurrentLoad is equal to the TripCapacity, the vehicle's CurrentDistance is equal to the total distance travelled from the depot to the customers and back to the depot, and the vehicle's CurrentRiskIndex is equal to the risk placeholder that was the lowest (*i.e.* either RiskSaver1A or RiskSaver1B). Furthermore, the two customers are added to the vehicle's GenerateSavingsRoute collection in the final chosen order (*i.e.* original or reversed orientation). If, however, any of the constraints are violated, the customer pair will not be added to a new route and, therefore, the next customer pair in the savings list is considered.

In the second case, the first customer in the pair (W1) is already included in some vehicle route, whereas the second customer in the pair (W2) is not. In this case, the internal boolean variable AssignedInSavingsRoute associated with customer W1 is equal to one, whereas the AssignedInSavingsRoute associated with customer W2 is zero. Customer W2 may, however, only be added to the same route as customer W1 if customer W1 is located on one of the two edges of the route (*i.e.* if customer W1 is either the first or last customer served on the route), because a customer may not be inserted in the middle of an existing route. If customer W1 is located on one of the edges of the route, the algorithm continues to assess whether customer W2 may be added to the same route without violating the user specified vehicle capacity limit (VehicleCapacity) constraint or the vehicle distance limit (DistanceConstraint) constraint. If these constraints are not violated, customer W2 may be added to the existing route in the case of the CVRP model (2.26)–(2.32). Therefore, if customer W1 is located at the start of the existing route, customer W2 is added before customer W1 in the existing route, as illustrated in Figure 5.10(a). If, however, customer W1 is located at the end of the existing route, customer W2 is added after customer W1 on the existing route, as illustrated in Figure 5.10(b).

(a) *Added to start.* (b) *Added to end.*

FIGURE 5.10: *Two possible route outcomes for the second case in the algorithm used to solve the CVRP model (2.26)–(2.32). In (a) customer* `w2` *is added to the start of the route before customer* `w1` *and in (b) customer* `w2` *is added at the end of the route after customer* `w1`*.*

In the case of the RCTVRP model (2.60)–(2.65), the algorithm defines two risk placeholders, denoted as RiskSaver2A and RiskSaver2B. The risk is calculated according to equation (2.59) and since the risk requires to be considered for both orientations of the route (as discussed in §5.2.2) the algorithm calculates four possible route outcomes (denoted as outcomes 2a–2d) for the route, as illustrated in Figure 5.11.

Case 2a and 2b. The first two routes for the second case is where customer W1 is located first in the existing route. Here, customer W2 is added in front of customer W1 and, therefore the

(a)
Route 2a: original orientation.

(b)
Route 2b: reverse orientation.

(c)
Route 2c: original orientation.

(d)
Route 2d: reverse orientation.

FIGURE 5.11: *Four possible route orientation outcomes for the second case in the algorithm. In (a) customer* `w2` *is added to the start of the route before customer* `w1` *and in (b) customer* `w2` *is added at the end of the route after customer* `w1`*. In (c) the route in (a) is reversed and in (d) the route in (b) is reversed.*

original route order is as illustrated in Figure 5.11(a) (*i.e.* route 2a) and the reversed route order is as illustrated in Figure 5.11(b) (*i.e.* route 2b). The possible risk for each of these orientations are calculated and stored in the risk placeholders named `RiskSaver2A` and `RiskSaver2B`, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold `MinimumRiskThreshold`. If `RiskSaver2A` achives the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the original route orientation is used. If, however, `RiskSaver2B` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the reversed route orientation is used.

The third and fourth routes for the second is where customer `W1` is located last in the existing route. Here, customer `W2` is added after customer `W1` and, therefore the original route order is as illustrated in Figure 5.11(c) (*i.e.* route 2c) and the reversed route order is as illustrated in Figure 5.11(d) (*i.e.* route 2d). The possible risk for each of these orientations are calculated and stored in the risk placeholders named `RiskSaver2A` and `RiskSaver2B`, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lowest risk does not exceed the user defined risk threshold `MinimumRiskThreshold`. If `RiskSaver2A` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the original route orientation is used. If, however, `RiskSaver2B` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the reversed route orientation is used.

It may be observed, however, that route 2a is the same as route 2d and that route 2b is the same as route 2d. Therefore, the location of customer `W1` in the existing route may be ignored in the RCTVRP model (2.60)–(2.65) and only routes 2a and 2b have to be considered. Once it is established that none of the constraints (*i.e.* capacity, distance or risk constraints) are violated in the second case, customer `W2` is added to the existing route by adding it to the vehicle's `GenerateSavingsRoute` collection, which contains the route the vehicle travels. In order to acknowledge the assignment of the customer to a route, the internal variable of `W2` is adapted so that the value of the boolean variable `AssignedInSavingsRoute` is equal to one. Furthermore, the vehicle agent's internal variables are adapted so that the vehicle's `CurrentLoad` is equal to the `TripCapacity`, the vehicle's `CurrentDistance` is equal to the total distance travelled from the depot to the customers and back to the depot, and the vehicle's `CurrentRiskIndex` is equal to the risk placeholder that was the lowest (*i.e.* either `RiskSaver2A` or `RiskSaver2B`). If, however, any of the constraints are violated, the

customer pair will not be added to a new route and, therefore, the next customer pair in the savings list is considered.

In the third case, the second customer in the pair (W2) is already included in some vehicle route, whereas the first customer in the pair (W1) is not. In this case the internal variable AssignedInSavingsRoute associated with customer W2 is equal to one, whereas the AssignedInSavingsRoute associated with customer W1 is equal to zero. Customer W1 may, however, only be added to the same route as customer W2 if customer W2 is located on one of the two edges of the route (*i.e.* if customer W2 is either the first or last customer served on the route), because a customer may not be inserted in the middle of an existing route. If customer W2 is located on the edge of the route, the algorithm therefore continues to assess whether customer W1 should be added to the same route without violating any user specified vehicle capacity limit (VehicleCapacity) or vehicle distance limit (DistanceConstraint) constraints. If these constraints are not violated, customer W1 may be added to the existing route in the case of the CVRP model (2.26)–(2.32). Therefore, if customer W2 is located at the start of the existing route, customer W1 is added before customer W2 in the existing route. If, however, customer W2 is located at the end of the existing route, customer W1 is added after customer W2 on the existing route. In the case of the RCTVRP model (2.60)–(2.65), however, the algorithm defines two risk placeholders, denoted as RiskSaver3A and RiskSaver3B. The risk is calculated according to equation (2.59) and since the risk has to be considered for both orientations of the route (as discussed in §5.2.2) the algorithm calculates four possible route outcomes as illustrated in Figure 5.12.

W1 → W2 → · · ·

(a)
Route 3a: original orientation.

· · · → W2 → W1

(b)
Route 3b: reverse orientation.

· · · → W2 → W1

(c)
Route 3c: original orientation.

W1 → W2 → · · ·

(d)
Route 3d: reverse orientation.

FIGURE 5.12: *Four possible route orientation outcomes for the third case in the algorithm. In (a) customer* `w2` *is added to the start of the route before customer* `w1` *and in (b) customer* `w2` *is added at the end of the route after customer* `w1`*. In (c) the route in (a) is reversed and in (d) the route in (b) is reversed.*

The first two routes in the third case is where customer W2 is located first in the existing route. Here, customer W1 is added before customer W2 and, therefore the original route order is as illustrated in Figure 5.12(a) and the reversed route order is as illustrated in Figure 5.12(b). The possible risk for each of these orientations are calculated and stored in the risk placeholders named RiskSaver3A and RiskSaver3B, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold MinimumRiskThreshold. If RiskSaver3A achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the original route orientation is used. If, however, RiskSaver3B achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the reversed route orientation is used.

The third and fourth routes in the third case is where customer W2 is located last in the existing route. In this case, customer W1 is added after customer W2 and, therefore the

original route order is as illustrated in Figure 5.12(c) and the reversed route order is as illustrated in Figure 5.12(d). The possible risk for each of these orientations are calculated and stored in the risk placeholders named RiskSaver3A and RiskSaver3B, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold MinimumRiskThreshold. If RiskSaver3A achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the original route orientation is used. If, however, RiskSaver3B achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the reversed route orientation is used.

It may be observed, however, that route 3a is the same as route 3d and that route 3b is the same as route 3d. Therefore, the location of customer W2 in the existing route may be ignored in the RCTVRP model (2.60)–(2.65) and only routes 3a and 3b have to be considered. Once it is established that none of the constraints (*i.e.* capacity, distance or risk constraints) are violated in the second route, customer W1 is added to the existing route by adding it to the vehicle's GenerateSavingsRoute collection, which contains the route the vehicle travels. In order to acknowledge the assignment of the customer to a route, the internal variable of customer W1 is adapted so that the value of the boolean variable AssignedInSavingsRoute is equal to one. Furthermore, the vehicle agent's internal variables are adapted so that CurrentLoad is equal to the TripCapacity, the vehicle's CurrentDistance is equal to the total distance travelled from the depot to the customers and back to the depot, and the vehicle's CurrentRiskIndex is equal to the risk placeholder that was the lowest (*i.e.* either RiskSaver3A or RiskSaver3B). If, however, any of the constraints were violated, the customer pair is not added to a new route and, therefore the next customer pair in the savings list is considered.

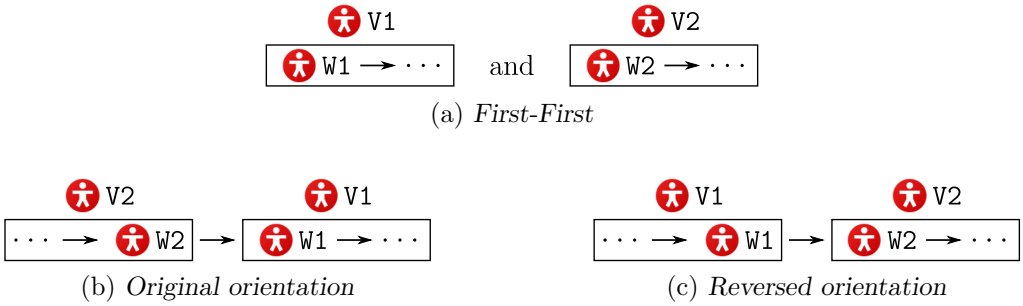
Furthermore, it may be noted that the core of the second case and the third case is the same. In both the second and third case, one customer is already included in the route, whereas the other is not. These two may, therefore, be regarded as the same.

In the final case, both of the customers, W1 and W2, have already been assigned to vehicle routes *i.e.* both cutsomers' internal variable AssignedInSavingsRoute has a boolean value equal to one. If customer W1 and customer W2 are assigned to the same vehicle route, or if either customer W1 or customer W2 are assigned somewhere in the middle of the route sequence, the pair is not be considered and the algorithm continues to consider the next customer pair in the savings list. Thus, the customer pair, W1 and W2, will only be considered if both customers are assigned to either the start or the end of two different vehicle routes. The algorithm, therefore runs through the list of vehicles that already have customers assigned to them, and determines which routes contain customers W1 and W2. There exists four possible outcomes for the CVRP model (2.26)–(2.32), namely first-first, first-last, last-first and last-last and there exists eight possible outcomes for the RCTVRP model (2.60)–(2.65) which include the four outcomes of the CVRP model (2.26)–(2.32) and their reversed alternatives. All of these possible route outcomes where previously listed in Table 5.3 and discussed in §5.2.2.

The two routes under consideration in the fourth case are referred to as route V1 and V2. In the case of the CVRP model (2.26)–(2.32) the two routes may only be linked into a single route if the combined capacity of the two routes do not exceed the user defined capacity limit (VehicleCapacity) and if the combined distance does not exceed the user defined distance limit (DistanceConstraint). In the case of the RCTVRP model (2.60)–(2.65), however, the algorithm includes the risk constraint and defines two risk placeholder variables, denoted as RiskSaver4A and RiskSaver4B. The risk is calculated according to equation (2.59) in §2.1.6 and since the risk has to be considered for both orientations of the route (as discussed

in §5.2.2) the algorithm calculates eight possible route outcomes (2 for each of the four cases discussed below) for the route.

In the first-first case, customer W1 is assigned first in route V1 and customer W2 is assigned first in route V2 as illustrated in Figure 5.13(a). The route may be linked by reversing route V2 and adding it to the start of route V1 (as illustrated in Figure 5.13(b)) and, therefore all of the customers from the GenerateSavingsRoute collection of route V2 is added to the GenerateSavingsRoute of route V1 and route V2 is deleted. The updated order of the GenerateSavingsRoute of route V1 will then serve as the original route orientation as illustrated in Figure 5.13(b), while the reversed version of this route is illustrated in Figure 5.13(c). The possible risk for each of these orientations are calculated and stored in the risk placeholders RiskSaver4A and RiskSaver4B, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold MinimumRiskThreshold. If RiskSaver4A achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the original route orientation is used. If, however, RiskSaver4B achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the reversed route orientation is used.



(a) *First-First*

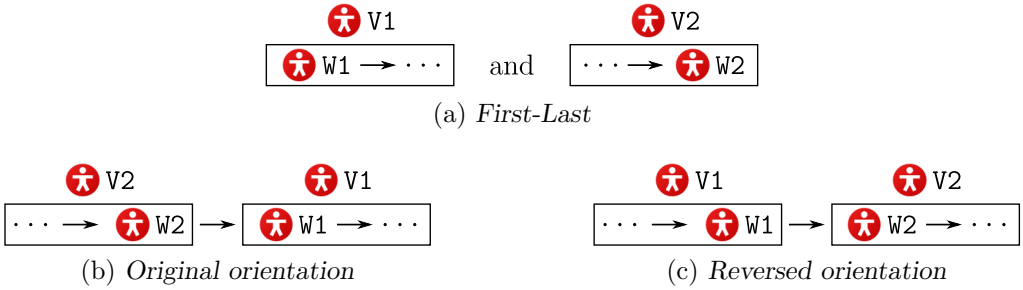
(b) *Original orientation*

(c) *Reversed orientation*

FIGURE 5.13: *Two possible route orientation outcomes for the first-first case in the fourth case in the algorithm. In (a) customer W1 is located at the start of route V1 and customer W2 is located at the start of route V2. In (b) route V2 is reversed and added to the start of route V1 and indicates the original route orientation, while in (c) the route in (b) is reversed.*

In the first-last case, customer W1 is assigned first in route V1 and customer W2 is assigned last in route V2 as illustrated in Figure 5.14(a). The route may be linked by simply adding route V2 at the start of route V1 in the order that it is and, therefore all of the customers from the GenerateSavingsRoute collection of route V2 is added to the start of the GenerateSavingsRoute of route V1 and route V2 is deleted. The updated order of the GenerateSavingsRoute of route V1 will then serve as the original route orientation as illustrated in Figure 5.14(b), while the reversed version of this route is illustrated in Figure 5.14(c). The possible risk for each of these orientations are calculated and stored in the risk placeholders RiskSaver4A and RiskSaver4B, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold MinimumRiskThreshold. If RiskSaver4A achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the original route orientation is used. If, however, RiskSaver4B achieves the lowest possible risk for the route and is also lower than the MinimumRiskThreshold, then the reversed route orientation is used.

In the last-first case, customer W1 is assigned last in route V1 and customer W2 is assigned first in route V2 as illustrated in Figure 5.15(a). The route may be linked by simply



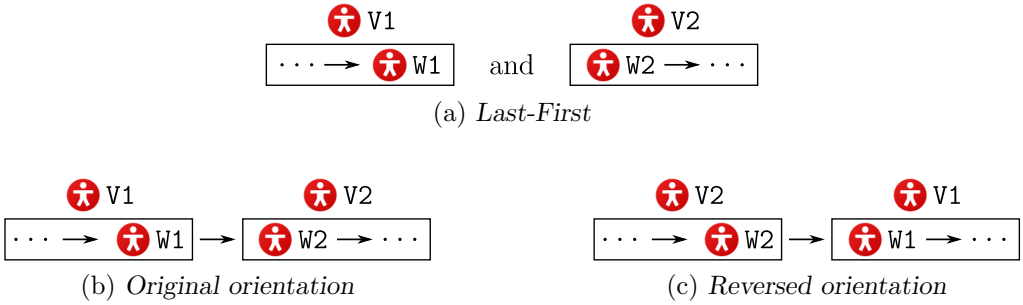
(a) *First-Last*

(b) *Original orientation*

(c) *Reversed orientation*

FIGURE 5.14: *Two possible route orientation outcomes for the first-last case in the fourth case in the algorithm. In (a) customer* `W1` *is located at the start of route* `V1` *and customer* `W2` *is located at the end of route* `V2`*. In (b) route* `V2` *is added to the start of route* `V1` *and indicates the original route orientation, while in (c) the route in (b) is reversed.*

adding route `V1` at the start of route `V2` in the existing order and, therefore all of the customers from the `GenerateSavingsRoute` collection of route `V1` is added to the start of the `GenerateSavingsRoute` of route `V2` and route `V1` is deleted. The updated order of the `GenerateSavingsRoute` of route `V2` will then serve as the original route orientation as illustrated in Figure 5.15(b), while the reversed version of this route is illustrated in Figure 5.15(c). The possible risk for each of these orientations are calculated and stored in the risk placeholders `RiskSaver4A` and `RiskSaver4B`, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold `MinimumRiskThreshold`. If `RiskSaver4A` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the original route orientation is used. If, however, `RiskSaver4B` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the reversed route orientation is used.



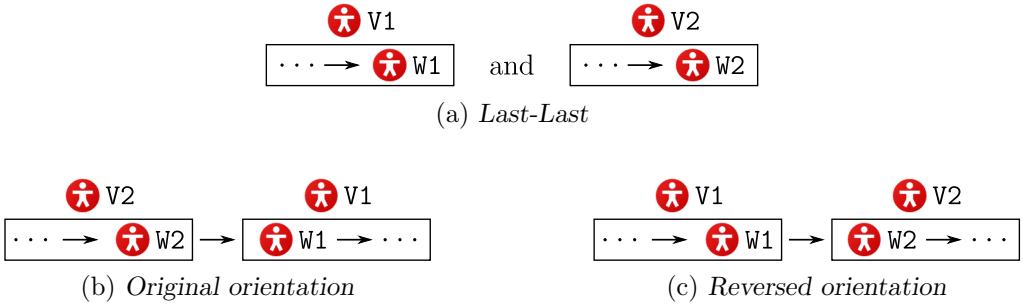
(a) *Last-First*

(b) *Original orientation*

(c) *Reversed orientation*

FIGURE 5.15: *Two possible route orientation outcomes for the last-first case in the fourth case in the algorithm. In (a) customer* `W1` *is located at the end of route* `V1` *and customer* `W2` *is located at the start of route* `V2`*. In (b) route* `V2` *is added to the end of route* `V1` *and indicates the original route orientation, while in (c) the route in (b) is reversed.*

In the last-last case, customer `W1` is assigned last in route `V1` and customer `W2` is assigned last in route `V2` as illustrated in Figure 5.16(a). The route may be linked by reversing route `V1` and adding it to the end of route `V2` and, therefore all of the customers from the `GenerateSavingsRoute` collection of route `V1` is added to the `GenerateSavingsRoute` of route `V2` and route `V1` is deleted. The updated order of the `GenerateSavingsRoute` of route `V2` will then serve as the original route orientation as illustrated in Figure 5.16(b), while the reversed version of this route is illustrated in Figure 5.16(c). The possible risk for each of these orientations are calculated and stored in the risk placeholders `RiskSaver4A` and



(a) *Last-Last*

(b) *Original orientation*

(c) *Reversed orientation*

FIGURE 5.16: *Two possible route orientation outcomes for the last-last case in the fourth case in the algorithm. In (a) customer* `W1` *is located at the end of route* `V1` *and customer* `W2` *is located at the end of route* `V2`*. In (b) route* `V2` *is reversed and added to the end of route* `V1` *and indicates the original route orientation, while in (c) the route in (b) is reversed.*

`RiskSaver4B`, respectively. The algorithm then compares the two risk values in order to establish which risk is lower (*i.e.* less risky) and the algorithm also determines if the lower risk does not exceed the user defined risk threshold `MinimumRiskThreshold`. If `RiskSaver4A` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the original route orientation is used. If, however, `RiskSaver4B` achieves the lowest possible risk for the route and is also lower than the `MinimumRiskThreshold`, then the reversed route orientation is used.

Once it is ensured that none of the constraints (*i.e.* capacity, distance or risk constraints) are violated in the fourth case, routes `V1` and `V2` are linked into a single route. In order to acknowledge the assignment, the vehicle agent' internal variables are adapted so that the vehicle's `CurrentLoad` is set equal to the combined `TripCapacity` for both routes that were joined, the vehicle's `CurrentDistance` is set equal to the total distance travelled from the depot to the customers and back to the depot, and the vehicle's `CurrentRiskIndex` is set equal to the risk placeholder that was the lowest (*i.e.* either `RiskSaver4A` or `RiskSaver4B`). If, however, any of the constraints were violated, the two routes will not be added into a single route, but will rather remain as is and the next customer pair in the savings list will be considered.

Once all the customers have been successfully assigned to routes, the algorithm terminates and the routes are returned as an output to the decision maker as decision support.

5.2.7 Visualisation in AnyLogic

After the "Run" button is selected and after the algorithm has established the routes, the model provides the second GUI to the user which is located in the `Main` agent and is shown in Figure 5.17. This GUI provides information in a box in the top left of the screen that describes the problem being executed. This information includes the number of depots (which in this model will always be equal to one), the number of customers that require service and the maximum capacity load a vehicle is allowed to carry. Furthermore, the GUI provides an interactive environment where the user is able to request certain actions and outputs. These outputs serve the purpose of validating anticipated results, rather than predicting results.

When the user is first presented with the screen, the graph information is not shown. The user is prompted to select the "1. Parameter evaluation" button, which when selected will run the parameter evaluation for the risk parameter. The parameter evaluation runs the same problem

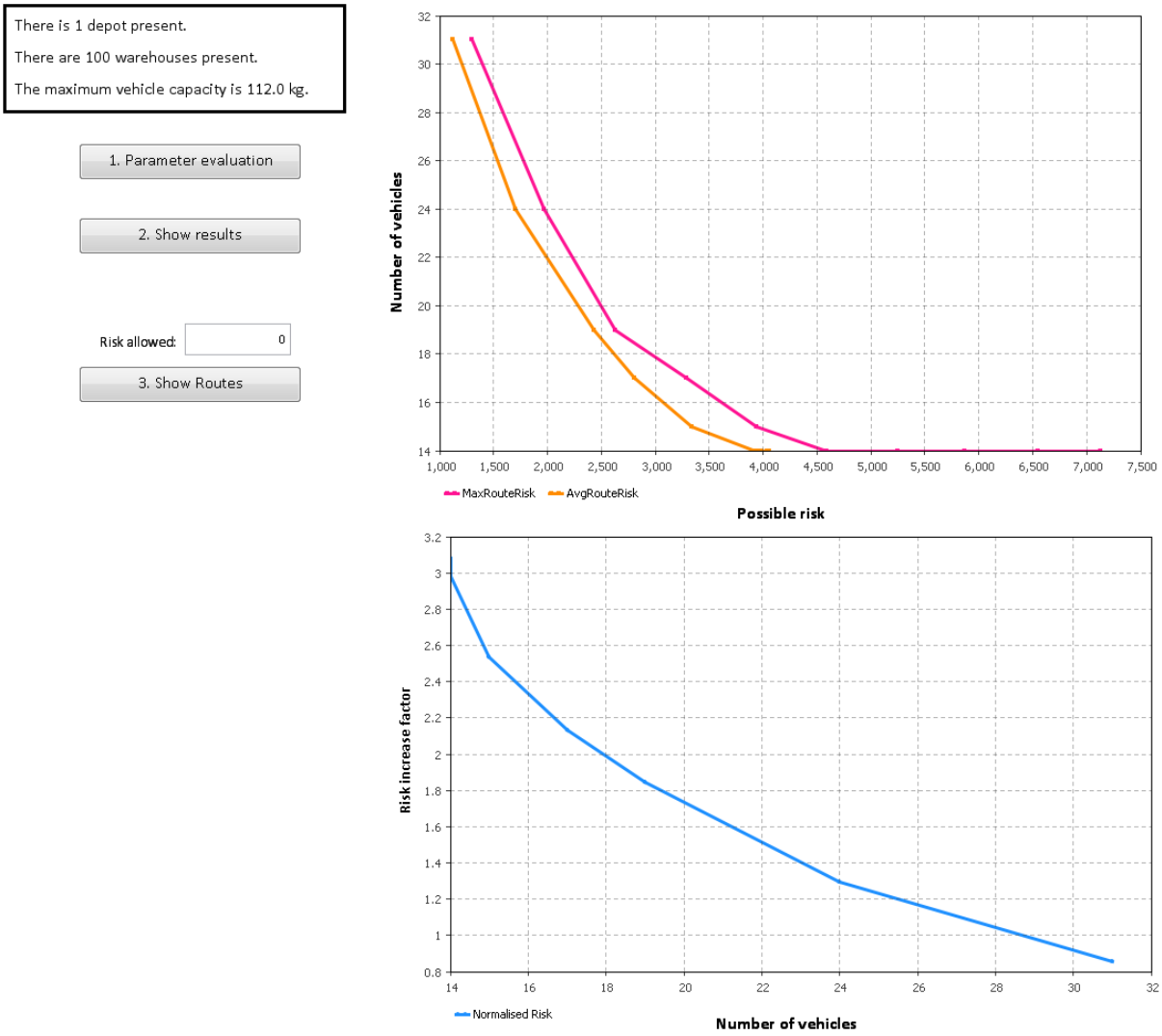


FIGURE 5.17: *The second GUI the user encounters in the model.*

instances for different risk thresholds in order to provide a variety of solutions to the user. The working of this parameter evaluation is discussed in more detail in §6.2.2.

Once the parameter evaluation is done, the user is able to select the "2. Show results" button, which will provide the user with the graphs as shown in Figure 5.17. The first graph shown in Figure 5.17 shows the user how the anticipated risk of the routes in the solution change depending on the number of `Vehicle` agents used. The x-axis denotes the possible (anticipated) risk associated with a route, while the y-axis denotes the number of `Vehicle` agents required to solve the problem at the given risk threshold. The pink line shows the risk of the route that incurred the most anticipated risk of all the routes in the solution (*i.e.* the most risky route), while the orange line shows the average risk incurred on the routes in the solution.

The second graph in Figure 5.17 shows the *risk increase factor* versus the number of `Vehicle` agents used to solve the problem. The x-axis denotes the number of `Vehicle` agents required to solve the problem at the given risk threshold, while the y-axis denotes the risk increase factor associated wit the number of `Vehicle` agents required to solve the problem. The second graph is an alternative illustration of the first graph in Figure 5.17, which may easily be interpreted by the user. The risk increase factor gives an indication of the % risk increase that is associated

with the number of `Vehicle` agents used to solve the problem. Thus, if the risk threshold is set to be at its strictest, then the risk increase factor will have to be less than or equal to 1 and then the maximum anticipated risk on any route in the solution is less than the minimum allowable risk threshold. If the risk increase factor is larger than 1, however, it means that the risk threshold needs to be made more lenient in order to allow the solution to use less `Vehicle` agents in order to solve the problem.

Once the user was able to analyse the results and solution options, the user may choose to enter the preferred route risk threshold of their choice in the text box in Figure 5.17, which is labelled "Risk allowed". This risk value may be read off of or interpreted from the first graph in Figure 5.17 according to the possible anticipated risk that the user is willing to accommodate, or the user may choose a risk threshold value according to the number of vehicles that the user is willing to deploy in order to solve the problem and effectively mitigate risk.

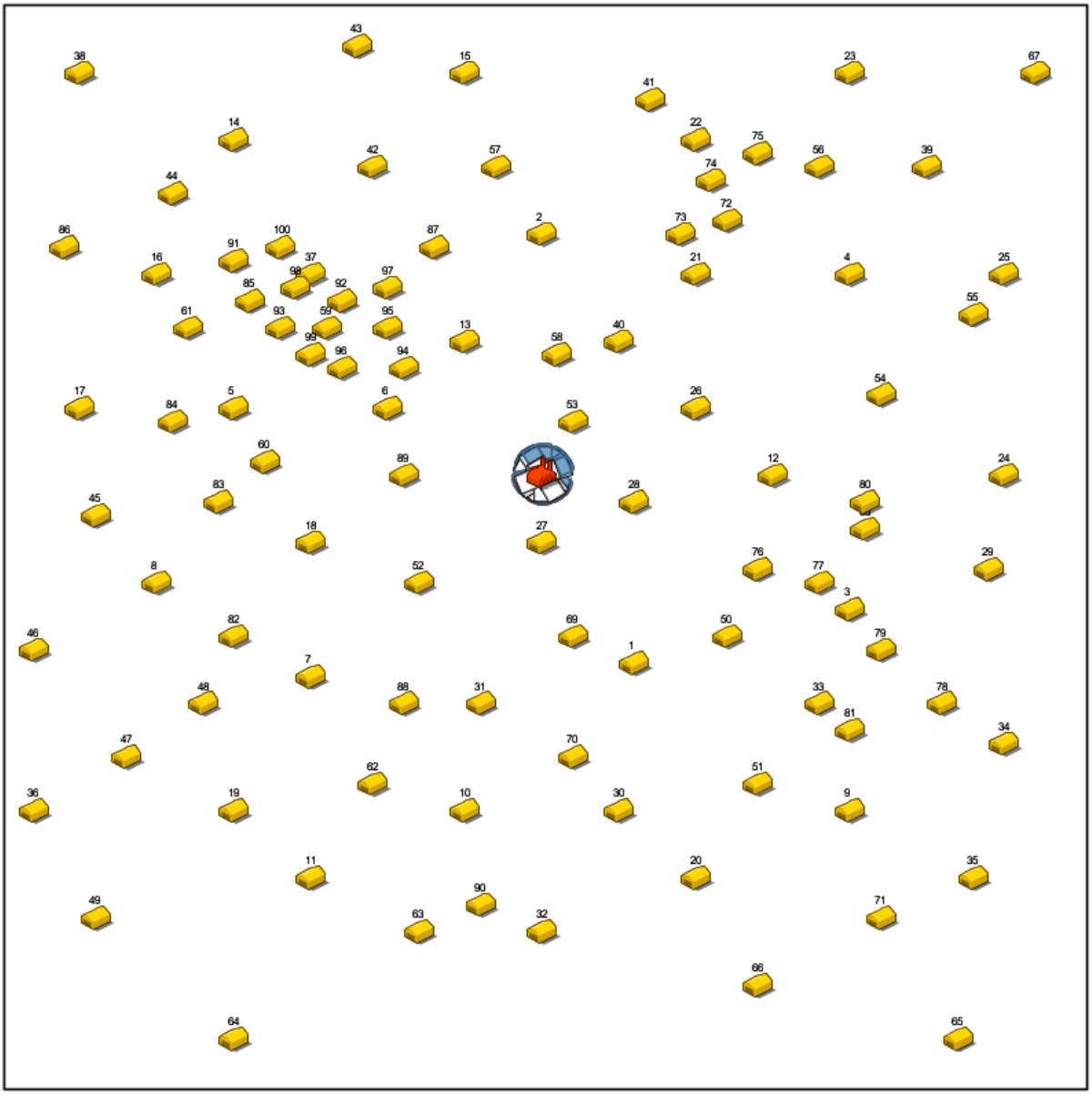


FIGURE 5.18: *A visual representation of the customer and depot layout.*

Once the user-defined risk threshold value is indicated in the edit box, the user may select the "3. Show routes" button. Upon the selection of this button, the model will determine the route layout that is associated with the user-preferred risk threshold value. The GUI then provides a visually animated representation of the `customer` agents in the network (indicated in yellow because they have not yet been serviced as of yet), the common `depot` agent, indicated in red, and the `vehicle` agents that are located at the depot, as illustrated in Figure 5.18. This interface enables the user to visualise the problem and gain a better understanding of the customer layout with reference to the depot, as well as what the typical routes might look like when executed.

Once the user selects the "3. Show Routes" button, the model will execute a visually animated representation of the route each `Vehicle` agent will travel — an example of such an execu-

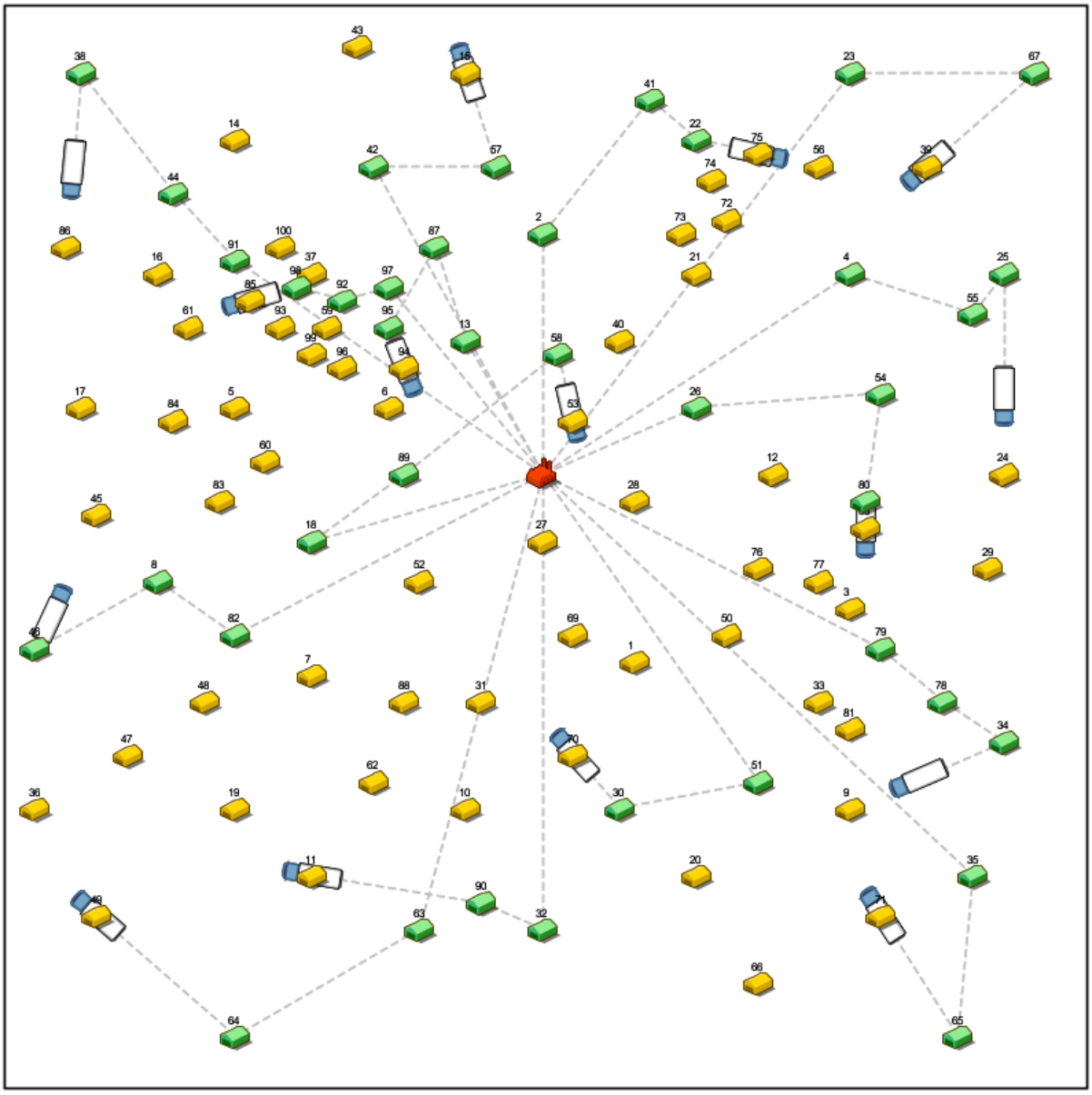


FIGURE 5.19: *The user is able to observe how the vehicle routes are executed in a visually animated manner through the GUI. The customers that require service are indicated in yellow, while the customers that have already been serviced are indicted in green.*

tion is illustrated in Figure 5.19. The route is illustrated by making use of a `TrackingLine` agent which is represented by the grey dashed line following each `Vehicle` agent. The `TrackingLine` agent operates in the "move" state of the `Vehicle` agent in the state chart that was illustrated in Figure 5.7 and discussed in §5.2.3. The `TrackingLine` agent determines the location of each `Vehicle` agent dynamically and follows each `Vehicle` along its route, while leaving a grey dashed line behind the `Vehicle` agent which represents a track that illustrates the path travelled by a `Vehicle` agent and also enables the user to visualise the previous destinations that were visited by the same vehicle. It may be observed that all of the customers that have a `TrackingLine` running through them are indicated in green — this means that the `Vehicle` agent that is associated with the specific `TrackingLine` has serviced these customers. The customers indicated in yellow, however, are still waiting on service.

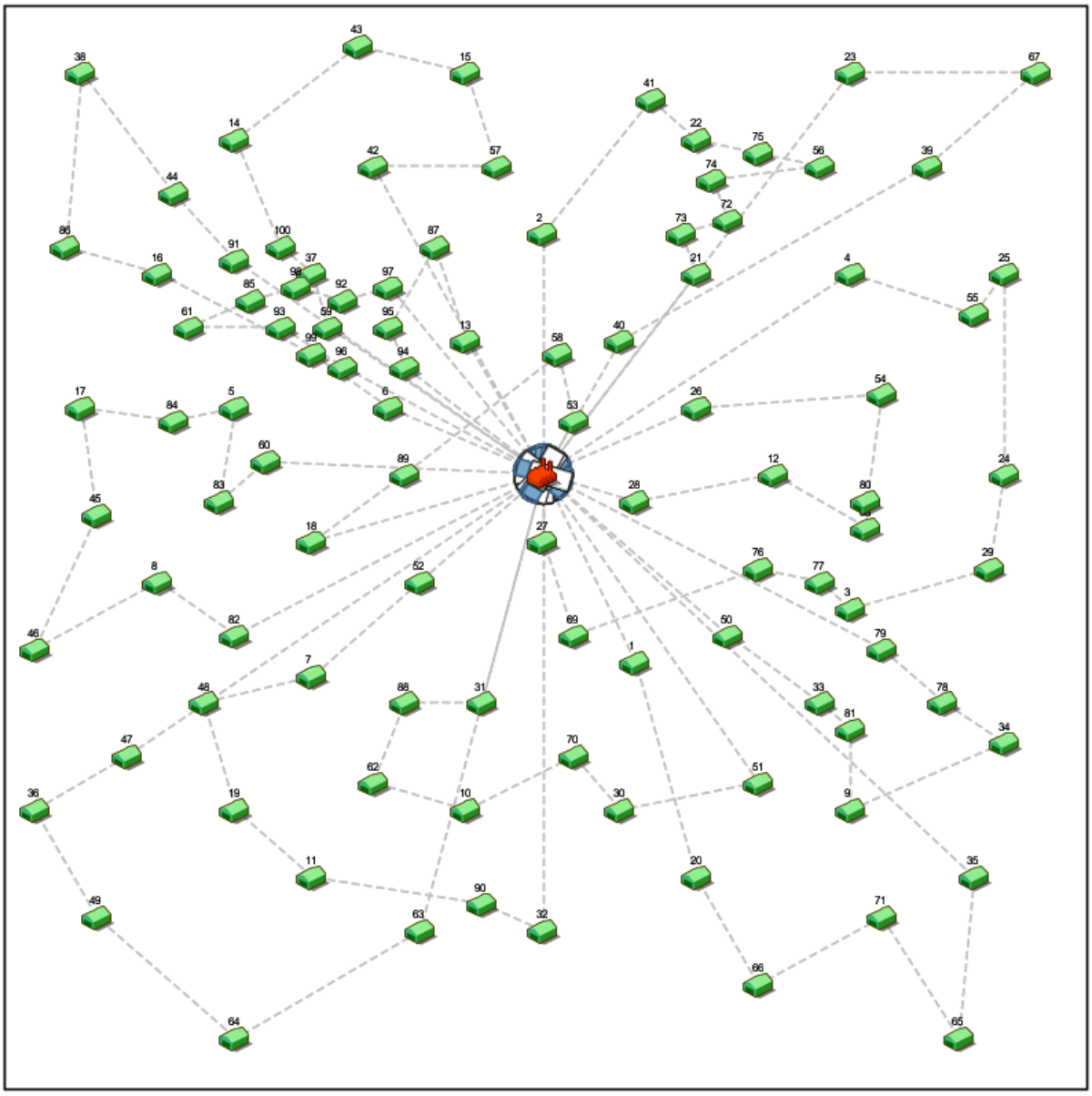


FIGURE 5.20: *The user is able to observe how the vehicle routes are executed to completion and, hence, the resulting routes may be visually observed.*

Finally, Figure 5.20 illustrates the GUI at the final time instance where all of the `Vehicle` agents have executed their respective routes and have returned back to the `depot` agent. It may be observed that all `Warehouse` agents are now indicated in green, which means that all of them have successfully been served by a `Vehicle` agent.

5.3 Chapter summary

The material that was covered in this chapter provided a sufficient foundation for the understanding of the working of the DSS model that was developed during this study. The chapter begun by discussing the modelling software that was used to solve the problem, as well as why this specific modelling software was chosen, and was discussed in §5.1. Furthermore, in section §5.1.2 the various features associated with the modelling software were listed and their respective functions were discussed. The chapter then continued, by explaining the proposed DSS framework in §5.2.1, where the various components that form part of the DSS were explained in detail. Section §5.2.2 discussed the working of the two algorithms that were implemented in the DSS's model base component (*i.e.* the CVRP algorithm and the RCTVRP algorithm) and provided the pseudo code for each algorithm and explained the working behind it. Furthermore, the chapter discussed the different object classes that form part of the developed model in §5.2.3. Object classes refer to the agents or objects in the model that are used to illustrate or perform certain tasks. In section §5.2.4, the working and visualisation of the graphical user interface were discussed. The GUI orchestrates various processes such as the model initialisation (which was discussed in §5.2.5), the optimisation of the algorithms in the ANYLOGIC modelling environment (which was discussed in §5.2.6) and, finally, the output results visualisation (which was discussed in §5.2.7). The chapter finally closed with a brief summary on the chapter contents.

CHAPTER 6

DSS model verification and validation

Contents

In this chapter the developed model along with the two implemented algorithms (CVRP and RCTVRP) are validated according to the methods previously discussed in Chapter 4.

Section 6.1 contains a detailed description of the methods that are used to verify whether the model was built according to the user-specified requirements. These methods include a top-down and modular development approach, accompanied by a variety of "checking" mechanisms. Furthermore, §6.2 contains a detailed description of the methods that are used to validate the model. The first method is a trace validation which is discussed along with examples in section §6.2.1. The second method of validation is a sensitivity analysis which is discussed in section §6.2.2. The sensitivity analysis is conducted in two parts. The first part of the sensitivity analysis aims to determine the computational robustness of the CVRP algorithm and is tested using test data from benchmark problems available in the literature. The second part of the sensitivity analysis aims to ascertain how a change in some parameters impact the performance and results provided by the RCTVRP algorithm. The chapter finally closes with a brief summary of the chapter contents in §6.3.

6.1 Model verification

Model verification was previously defined in §4.4.2 as the method of affirming that the model structure is built correctly (*i.e.* whether the model works according to the specified user requirements). The model described in §5.2 was developed in a top-down and modular manner which allowed the developer to focus on smaller parts of the system while keeping the bigger picture and goal in mind. The author, therefore, repeatedly compiled and executed the model after each new addition in the program code in order to ensure its working. Each individual section of the model represents a vital part of the working of the model and, therefore, each module was continuously iterated and checked for errors before continuing to add model complexity.

Verification was continuously performed throughout the model development by implementing various checking mechanisms. One such checking mechanism is the *trace function* provided by ANYLOGIC in the form of a *system out print* function. This function may be implemented anywhere in the software code by the developer in order to *trace* or *flag* variables through the system. This function allows the developer to keep track of variables and how they change through various iterations, by providing a "system out print" output to an internal ANYLOGIC console. The console is an internal information display window in ANYLOGIC that displays output information that is either prompted by the user or that are triggered by errors in the program code. From the implementation of the trace function at various different steps in the algorithm code and the analysis of the "system out print" outputs in the console, it was possible to draw a conclusion on the verification of the model. The first conclusion is that the model is configured correctly and according to the input parameters provided by the user. The second conclusion is that the model does not violate any constraints when constructing vehicle routes and the last conclusion is that the model runs through the implemented algorithms in a logical manner.

Another checking mechanism is the output display provided through the developed GUI. The GUI provides a clear visual representation of the problem and of the results to the user. The user is able to visually analyse the routes and graphs associated with the problem's solution output and the user may therefore easily draw conclusions from the visual medium. From the GUI output, it was possible to confirm that the model does in fact service a network of individual customers with a single common depot. This was determined by observing that the model displays a number of `Warehouse` agents that are equal to the number of customers in the user-defined network and that there is a single `Depot` agent present. Furthermore, once the routes are shown, it may be observed that a fleet of vehicles start and end their respective routes at the common depot and that each customer is served exactly once by a single vehicle. Furthermore, the model output serves as a validation of the expected outcome, rather than a prediction of a certain outcome. Thus, the user-requirements are met without violating any constraints.

Finally, the built in de-bugging functions and error indicators of the ANYLOGIC modelling software (also known as *interactive run controllers* (IRCs)) assisted in acknowledging errors in the code as they arise. Once errors are identified, ANYLOGIC provides a detailed description of the error and its location in the program's output terminal. In some cases, the program may encounter errors in the midst of an iteration run and then the run is suspended until the errors are fixed and the program may continue to run through the code again. The developer was therefore able to easily address these errors and bugs accordingly throughout the entire development process. This checking mechanism was vital in the development of the model since it allowed the user to test the logic and syntax of each new component as it was introduced to the model in a modular manner. The modular approach combined with the use of IRCs therefore prevents the model from advancing too quickly and allowing errors to accumulate, since the accumulation of errors tend to make the de-bugging process even more complex.

6.2 Model validation

Model validation was previously defined §4.4.2 as the method of confirming that a model is indeed developed to operate within a satisfactory range of accuracy. In order for a model to be considered as "valid", it must exhibit a level of reasonableness according to five factors, namely continuity, consistency, degeneracy, nonsensical conditions and extremes (as discussed in §4.4.2). The validation of this model relies mainly on validation testing through the use of empirical data,

otherwise referred to as test data. According to research done by Martis [87], the value of a model increases as the confidence level of the model increases. The confidence level will only increase if the developed model achieves similar results to the results achieved in the test data. The validation of the model, therefore serves as a confirmation of whether or not the results from the developed model are a good representation of the target results as represented by the test data. In this particular study, the target is therefore to apply the formulation of a RCTVRP based on the Clarke-Wright savings algorithm to a set of customers in order to suggest suitable vehicle routes that mitigate risk.

Two of the validation methods listed in Table 4.2 were employed in order to gain insight as to how accurate the developed model is. The first method is *tracing*, which tracks specific entities throughout the model execution in order to observe their behaviour and to identify any nonsensical outputs provided by the model, while the second method is a *sensitivity analysis* that observes the effect that varying input parameters have on the model output results and thereby validates the model's reasonableness. Both of these methods are discussed in further detail in the remainder of this section.

6.2.1 Trace validation

Trace validation is used to track the behaviour of specific entities in a model. In this particular model the specific entities that require tracking are the `Vehicle` agents. The `Vehicle` agents are required to perform a series of movements according to its state chart (as previously shown in Figure 5.7). Furthermore, the route travelled by the `Vehicle` agent is optimised by the selected algorithm (either CVRP or RCTVRP) and the algorithm populates the necessary number of `Vehicle` agents. In order to ensure that the algorithm executes correctly and that the `Vehicle` agent follows the correct movements, some trace elements (or flags) may be implemented. In this case, traces in the form of "system out print" functions were used in order to check the program logic at any given moment during the execution run. These "system out print" functions have the ability to indicate the value of certain variables at specific time instances during the model's execution run. The output is prompted by the user by inserting the function in various parts of the program code in order to keep track of various elements.

In the case of the RCTVRP, it is important to keep track of variables such as the vehicle's current demand (`CurrentLoad`), the current distance travelled (`CurrentDistance`) and the current risk incurred by the vehicle (`CurrentRiskIndex`) in order to ensure that the distance, capacity and risk threshold constraints are not violated. Furthermore, the logic of the algorithm may be analysed in order to establish whether the algorithm executes as it is expected to. Figure 6.1 and Figure 6.2 provide examples of what a trace may look like when implemented in the program code, while Figure 6.3 illustrates what the system output of these traces may look like in the AnyLogic console. Figure 6.1 tracks the occurrence of a "First case" scenario (as explained in §5.2.6) in the algorithm logic, while Figure 6.2 tracks the occurrence of a "Fourth case – First-First" scenario (as explained in §5.2.6) in the algorithm logic.

```
//if neither have been put in a route AND not exceed demand AND not exceed risk
if(Warehouse1.AssignedInSavingsRoute==false && Warehouse2.AssignedInSavingsRoute==false && ((Warehouse1.Demand+Warehouse2.Demand)<VehicleCapacity))
{
    System.out.println(" ");
    System.out.println(" -------------------------- ");
    System.out.println("FIRST CASE - NEITHER:  " + "Warehouse " + Warehouse1.WarehouseName + ", Warehouse " + Warehouse2.WarehouseName);
```

FIGURE 6.1: *Example of the trace function implemented in the program code in order to track the occurrence of a "first case" scenario in the algorithm.*

```
//if the FIRST warehouse IN EACH route is the warehouses we are considering
if(V1.GeneratedSavingsMethodRoute.getFirst() == Warehouse1 && V2.GeneratedSavingsMethodRoute.getFirst() == Warehouse2)
{

    System.out.println(" ");
    System.out.println(" ---------------------------- ");
    System.out.println("FOURTH CASE:  " + "Warehouse " + Warehouse1.WarehouseName + ", Warehouse " + Warehouse2.WarehouseName);
    System.out.println("Instance A occurred:");
    System.out.println("V1 is " + V1.getIndex() + " with customers " + V1.GeneratedSavingsMethodRoute.size() + ", V2 is " + V2.getIndex()
    System.out.println("V1 current capacity is " + V1.CurrentLoad + ", V2 current capacity is " + V2.CurrentLoad);
```

FIGURE 6.2: *Example of the trace function implemented in the program code in order to track the occurrence of a "fourth case – first-first" scenario in the algorithm.*

```
Console
anylogic config [Java Application] C:\Program Files\AnyLogic 8 Personal Learning Edition\jre\bin\javaw.exe (06 Nov 2018, 17:39:03)
 I T E R A T I O N  13
 Risk threshold is 539712.256670163

 ----------------------------
FIRST CASE - NEITHER:  Warehouse 1.0, Warehouse 2.0

 ----------------------------
FIRST CASE - NEITHER:  Warehouse 3.0, Warehouse 4.0

 ----------------------------
SECOND CASE - part A happend (w1 first): Warehouse 1 is 2.0 and warehouse 2 is 5.0

 ----------------------------
FOURTH CASE:  Warehouse 1.0, Warehouse 3.0
Instance A occurred:
V1 is 0 with customers 3, V2 is 1 with customers 2
V1 current capacity is 3900.0, V2 current capacity is 2200.0
```

FIGURE 6.3: *Example of the system output results in the console during iteration thirteen of the algorithm execution where there were two occurrences of the "first Case" and one occurrence of the "fourth Case – first-first.*

This method of validation may, for example, be used to compare the occurrence of these different scenarios with the actual entries in the savings list in order to determine whether the algorithm did indeed consider each pair of customers in the savings list and also whether the pairs were interpreted correctly with regards to the "case" it is classified as. The method of validation may also be used to verify whether a specific customer has been assigned to a vehicle route or not, as well as how many vehicles are present in the model at any given time. Furthermore, these trace statements may be inserted in the code of the vehicle state chart in order to observe which vehicles are in which state at any given time and also to identify any nonsensical outputs that may occur in the model.

The implementation and analysis of these trace functions have established that the algorithm does in fact work as it is required to in the modelling environment and that the model is reasonable. The model is able to identify nonsensical inputs and display error messages accordingly, instead of providing nonsensical outputs. The model is also consistent in the approach followed and output results.

6.2.2 Sensitivity analysis

The sensitivity analysis performed in this project on the models of §5.2.2 aim to achieve two goals. The first goal aims to determine the computational robustness of the algorithm that is implemented in the developed model and the second goal aims to ascertain how the change in some parameters impact the performance and results provided by the algorithm that is implemented in the developed model.

The sensitivity analysis was conducted by employing two different methods. The first method was to compare the developed model with *test data* in the form of benchmark problems from

the literature in order to test the model's consistency, degeneracy and continuity (as discussed in §4.4.2) with regards to the CVRP algorithm. In this analysis, the objectives mentioned previously are measured with specific reference to the total distance travelled by the vehicles that form part of the solution. The second method in which the sensitivity analysis was performed was through a *parameter variation* in order to test the model's extremes (also discussed in §4.4.2) with regards to the RCTVRP algorithm. In this analysis, the objectives mentioned previously are measured with specific reference to the maximum risk incurred on a single vehicle route travelled by one of the vehicles that form part of the solution. Both of these methods of sensitivity analysis are described in the remainder of this section.

Test data

The developed CVRP model was tested using test data sets from the VRP benchmark library [63] that is available online. The benchmark problems in the VRP library provide the locations of customers in a set along with their respective demands and the vehicle's capacity limit. Each problem scenario also provides a solution solved optimally, which indicates the vehicle routes along with each route's respective demand on board and the distance travelled by each vehicle. Since the RCTVRP algorithm is based on the CVRP, the CVRP algorithm had to be validated before the RCTVRP algorithm was developed.

There exists many benchmark test data sets in the VRP library for the CVRP. The developed model was tested using ten data sets from the VRP library named as *Christofedes and Eilon (Set E)*. The ten benchmark problems used for this validation are listed in Table 6.1. The table provides the name of the test data set in the left-most column along with the optimal solution of the benchmark problem and the results achieved by the developed CVRP model, using the Clarke-Wright savings algorithm. The number of nodes (*i.e.* the number of customers and the depot) are denoted by n, while the number of vehicles used in each instance is denoted by k, the total distance travelled by all vehicles are denoted by d. The total capacity limit of each vehicle is denoted by L and the average capacity on board each vehicle in the Clarke-Wright savings model is denoted by avg L. Furthermore, the similarity of the Clarke-Wright savings model associated with the benchmark model is denoted by %. The similarity provides an indication of the accuracy of the developed model. The similarity may be calculated as [148]

$$\text{similarity} = 1 - \%\ (\text{error}), \tag{6.1}$$

$$= 1 - \left(\frac{\text{experimental} - \text{theoretical}}{\text{theoretical}}\right). \tag{6.2}$$

In this case, the "experimental" value refers to the total distance travelled by all vehicles in the calculated model using the Clarke-Wright savings algorithm, while the "theoretical" value refers to the total distance travelled by all the vehicles in the optimal benchmark model solution. If the similarity value is less than one it means that the "experiment value" performed worse than the "theoretical" value, while if the similarity value is more than one it means that the "experiment value" performed better than the "theoretical" value.

The purpose of this validation is to check that the Clarke-Wright algorithm applied to the CVRP problem does produce vehicle routes that are constrained by the vehicle's capacity limit, while attempting to travel the shortest possible distance in order to serve all the customers in the network.

			BM		CW			
Problem instance	n	L	k	d	k	d	avg L	%
E_n22_k4	22	6 000	4	375	4	389	5 625	0.96
E_n23_k3	23	4 500	3	569	3	621	3 396	0.91
E_n30_k3	30	4 500	3	534	4	520	3 188	1.03
E_n33_k4	33	8 000	4	835	4	845	7 343	0.99
E_n51_k5	51	160	5	521	6	857	130	0.87
E_n76_k7	76	220	7	682	7	747	195	0.90
E_n76_k8	76	180	8	735	8	799	171	0.91
E_n76_k10	76	140	10	830	10	900	136	0.92
E_n101_k8	101	200	8	815	8	890	182	0.91
E_n101_k14	101	112	14	1067	14	1 139	104	0.93
							Avg %	0.93

TABLE 6.1: *The results of the Clarke-Wright savings algorithm compared to the optimal results achieved in the ten benchmark problems from the VRP library [63].*

Research by Martis [87] states that validation should include the identification and quantification of errors in conceptual models (*i.e.* the developed model), while the accuracy of the conceptual model is subsequently measured with relation to the empirical or test data (which serves as the best possible measure of reality that is available). From Table 6.1 it may be observed that the Clarke-Wright algorithm performs as expected. Since the Clarke-Wright algorithm is classified as a simple heuristic, it achieves good results that are not necessarily optimal. If the % similarity is considered, however, the Clarke-Wright savings algorithm performs relatively well when compared to the results achieved in the benchmark problems. There are two cases where the Clarke-Wright savings algorithm allowed for more vehicles to be used in order to serve the customers *i.e.* in case E_n30_3 and case E_n51_5. In the case of E_n30_3 the Clarke-Wright savings algorithm used one more vehicle than in the benchmark model, however, the distance travelled in the Clarke-Wright savings algorithm is less than in the benchmark model. In the case of E_n51_5, however, the Clarke-Wright savings algorithm used one more vehicle than in the benchmark model and the distance travelled by the Clarke-Wright savings algorithm is more than in the benchmark model. Both of these outliers were inspected further in order to ensure that their working is correct and it was indeed proven to be correct. Research by Martis [87] supports this inaccuracy, by stating that a model will hardly ever be absolutely valid, however, it should be valid for the purpose for which it is constructed.

When considering the reasonableness of the developed model, it may be observed that the developed model displays signs of continuity and is consistent in similar runs and does not display signs of degeneracy (*i.e.* changes to the input parameters, such as capacity constraints, have the same limiting effects on the output results in all problem instances). It was, however, found that solutions became more difficult to solve as the problems became larger and when the problems had small demand variations between the customers. All test instances did, however, achieve good solutions very fast. Overall, it may be concluded that the concept of the Clarke-Wright savings algorithm and its implementation in the context of a CVRP was applied accurately in the developed model.

Parameter variation

Since the working of the RCTVRP is based on that of the CVRP, the capacity of the RCTVRP model is already validated through the validation method described in the previous section. The

purpose of this part of the parameter variation is, therefore, to observe the effect of different risk threshold values on a single problem instance. Since there does not exist a RCTVRP library with test instances in the literature, existing test data sets from the benchmark VRP library [63] were adapted and used as RCTVRP benchmark test data sets. As previously explained in §2.1.6, a RCTVRP library may be constructed by defining a minimum risk threshold (T) for each problem instance by establishing $T = max_{i \in N}\{d_i \times c_{i0}\}$ for the problem. This minimum allowable risk threshold may then be made more lenient by multiplying it with an incremental risk factor.

The RCTVRP model was tested using thirteen different data sets from the VRP library named as *Christofedes and Eilon (Set E)*. The first ten sets are the same as in Table 6.1. These thirteen data sets from the benchmark VRP library [63] were combined with twenty nine different risk threshold levels. These risk threshold levels were populated by multiplying the minimum allowable risk threshold, denoted as $RL1$, by an increasing factor of 0.5 (as described in §2.1.6), with a lower bound of one and an upper bound of twenty nine. The lower and upper bounds were chosen as one and twenty nine, respectively, because from the complete data as shown in Appendix A, it may be observed that the number of vehicles do not decrease after a certain risk threshold and, therefore, it is not necessary to make the risk threshold more lenient (*i.e.* make the higher bound more than twenty nine). The different risk thresholds are illustrated in Table 6.2.

Risk threshold (T)	**Value**
$RL1$	$max_{i \in N}\{d_i \times c_{i0}\}$
$RL1.5$	$1.5 \times RL1$
$RL2$	$2.0 \times RL1$
$RL2.5$	$2.5 \times RL1$
$RL3$	$3.0 \times RL1$
$RL3.5$	$3.5 \times RL1$
$RL4$	$4.0 \times RL1$
$RL4.5$	$4.5 \times RL1$
$RL5$	$5.0 \times RL1$
$RL5.5$	$5.5 \times RL1$
$RL6$	$6.0 \times RL1$
$RL6.5$	$6.5 \times RL1$
$RL7$	$7.0 \times RL1$
$RL7.5$	$7.5 \times RL1$
$RL8$	$8.0 \times RL1$
$RL8.5$	$8.5 \times RL1$
$RL9$	$9.0 \times RL1$
$RL9.5$	$9.5 \times RL1$
$RL10$	$10.0 \times RL1$
$RL10.5$	$10.5 \times RL1$
$RL11$	$11.0 \times RL1$
$RL11.5$	$11.5 \times RL1$
$RL12$	$12.0 \times RL1$
$RL12.5$	$12.5 \times RL1$
$RL13$	$13.0 \times RL1$
$RL13.5$	$13.5 \times RL1$
$RL14$	$14.0 \times RL1$
$RL14.5$	$14.5 \times RL1$
$RL15$	$15.0 \times RL1$

TABLE 6.2: *Twenty nine different risk threshold levels used in the parameter evaluation.*

A summary of the two extremes of the output results and their corresponding observations are shown in Table 6.3. The table shows the results of the strictest risk threshold (*i.e.* the minimum

allowable risk threshold where $T = RL1$) and of the most lenient risk threshold (*i.e.* where $T = RL15$). The table provides the name of the data set in the left-most column. The number of nodes (*i.e.* the customers and the depot) are denoted by n, while the number of vehicles used to solve the problem is denoted by k and the vehicle's capacity limit is denoted by L. The minimum allowable risk threshold that is defined for each problem is denoted by min T.

					Strict ($RL1$)				Lenient ($RL15$)				
Problem instance	n	L	k	min T	k	d	avg L	% R	k	d	avg L	avg R	%
E_n22_k4	22	6000	4	77102	10	682	2250	1.00	4	389	5625	1.76	0.24
E_n23_k3	23	4500	3	137457	5	770	2038	1.00	3	621	3396	1.74	0.26
E_n30_k3	30	4500	3	174042	5	679	2550	1.00	4	520	3188	1.88	0.12
E_n33_k4	33	8000	4	357816	8	1469	3671	1.00	4	845	7343	1.81	0.19
E_n51_k5	51	160	5	909	22	1338	35	1.03	6	600	130	1.23	0.80
E_n76_k7	76	220	7	1341	28	1756	49	1.00	7	740	195	1.21	0.79
E_n76_k8	76	180	8	1341	28	1756	49	1.00	8	699	171	1.36	0.64
E_n76_k10	76	140	10	1341	28	1756	49	1.00	10	900	136	1.52	0.48
E_n101_k8	101	200	8	1318	31	2080	47	1.01	8	890	182	1.23	0.78
E_n101_k14	101	112	14	1318	31	2080	47	1.01	14	1139	104	1.64	0.38
E_n121_k7	121	200	7	1990	37	5366	37	1.00	7	1098	196	1.30	0.71
E_n151_k12	151	200	12	1318	47	2988	47	1.00	12	1155	186	1.37	0.63
E_n200_k16	200	200	16	1334	64	3917	49	1.00	17	1419	184	1.22	0.78

TABLE 6.3: *The results of the parameter variation performed on the RCTVRP algorithm for the two extreme cases i.e. the strictest risk threshold and the most lenient risk threshold.*

For the strictest variation of the risk threshold (*i.e.* $RL1$), the model produces vehicle routes subject to a strict risk threshold which limits the possible risk encountered on a single vehicle route to the minimum allowable risk. The total distance travelled by all the vehicles in the solution is denoted by d and the average load carried on each vehicle in the solution is denoted by avg L. Furthermore, the maximum possible risk encountered along a vehicle route should be less than or equal to the minimum allowable risk threshold and is calculated with the similarity equation in (6.1) and is denoted by % R. If % $R \leq 1$ the maximum risk encountered on a single route in the solution is more than the risk threshold and if % $R \geq 1$ the maximum risk encountered on a single route in the solution is less than the risk threshold.

For the most lenient variation of the risk threshold (*i.e.* $RL15$), the model produces vehicle routes subject to a more lenient risk threshold which limits the possible risk encountered on a single vehicle route to $RL15$. The total distance travelled by all the vehicles in the solution is denoted by d and the average load carried by each vehicle in the solution is denoted by avg L. Furthermore, the maximum possible risk encountered along a vehicle route should be less than or equal to $RL15$ and is calculated with the similarity equation (6.1) and is denoted by % R. Finally, in the right-most column the most lenient variation of the routes and the strictest variation of the routes are compared to see what the relative difference in encountered risk is and is denoted by %.

The relative difference in risk encountered is calculated using the similarity equation, where

$$\text{relative difference} = 1 - \left(\frac{\text{experimental} - \text{theoretical}}{\text{theoretical}}\right), \tag{6.3}$$

$$= 1 - \left(\frac{\text{lenient \% } R - \text{strict \% } R}{\text{strict \% } R}\right). \tag{6.4}$$

If the relative difference is less than zero ($\% \leq 0$) it means that the risk increased from the strictest case to the more lenient case, whereas if the relative difference is more than zero ($\% \geq 0$) it means that the risk decreased from the strictest case to the most lenient case

From Table 6.3, it may be observed that the RCTVRP algorithm performs as expected. In the results obtained by the strictest variation of the routes (RL1), it may be observed that $\% \ R \geq 1$ for all test instances. Therefore, the maximum risk encountered on a single vehicle route is less than or equal to the risk threshold applied to the problem. It may also be noted that the average demand on each vehicle (avg L) is much lower than the actual vehicle capacity L. This is favourable since it has previously been stated in §2.1.6 that less variables on board the vehicle incurs a lower possible risk, because the loss of valuable goods would be less in the case that a heist occurs. In the results obtained by the most lenient variation of the routes (RL15), it may be observed that $\% \ R \geq 1$ for all test instances. Therefore, the maximum risk encountered on a single vehicle route is less than or equal to the risk threshold applied to the problem. In this case, however, it may also be noted that the average demand on each vehicle (avg L) is closer to the actual vehicle capacity L than it was in the strict variation and therefore the loss of valuable goods during a heist would be more than in the case of the strict risk threshold.

When considering the reasonableness of the model it may, therefore, be observed that the model works correctly at its extreme conditions (*i.e.* at a strict risk threshold or at a lenient risk threshold). Overall, it may be concluded that the concept of the modified Clarke-Wright savings algorithm and its implementation in the context of a RCTVRP was applied accurately in the developed model.

In order to observe how much the individual risk of each route increases for each new risk threshold level (in Table 6.2) that is enforced on the problem, a risk increase factor had to be calculated. In doing so, the individual route that accumulated the most anticipated risk for the specific problem instance solution (*i.e.* maximum route risk) was compared to the minimum allowable risk threshold (*i.e.* $RL1$) for each problem instance. Thus, the risk increase factor was calculated as

$$\text{risk increase factor} \ = \ \frac{\text{maximum route risk}}{\text{minimum allowable risk threshold}}. \tag{6.5}$$

The risk increase factor results for the *maximum risk on a single route* for each of the thirteen problem instances listed in Table 6.3 combined with the twenty nine different risk levels are illustrated graphically in Figure 6.4.

The x-axis denotes the number of vehicles required to solve the problem without violating any constraints, while the y-axis denotes the risk increase factor that may be expected if the risk threshold level is made more lenient. The red line indicates the minimum allowable risk threshold ($RL1$) for each problem instance (*i.e.* risk increase factor $=$ 1). It may be observed that all the data points on the red line are the cases where the risk of the routes are the most strictly constrained and, therefore the risk increase factor is equal to 1. For example, when analysing E_n200_k16 it may be observed that when the risk threshold is at its strictest (*i.e.* equal to $RL1$) the solution requires 64 vehicles and the route with the maximum anticipated risk will be equal to a factor one (*i.e.* equal to the minimum allowable risk threshold), however, when the risk threshold is at the most lenient (*i.e.* equal to $RL15$) the solution requires seventeen vehicles and the maximum anticipated risk will increase by a factor 11.7 (more than the minimum allowable risk threshold).

Similarly, the risk increase results for the *average risk for the fleet of vehicles* used for each of the thirteen problem instances listed in Table 6.3 combined with the twenty nine different risk levels are illustrated graphically in Figure 6.5.

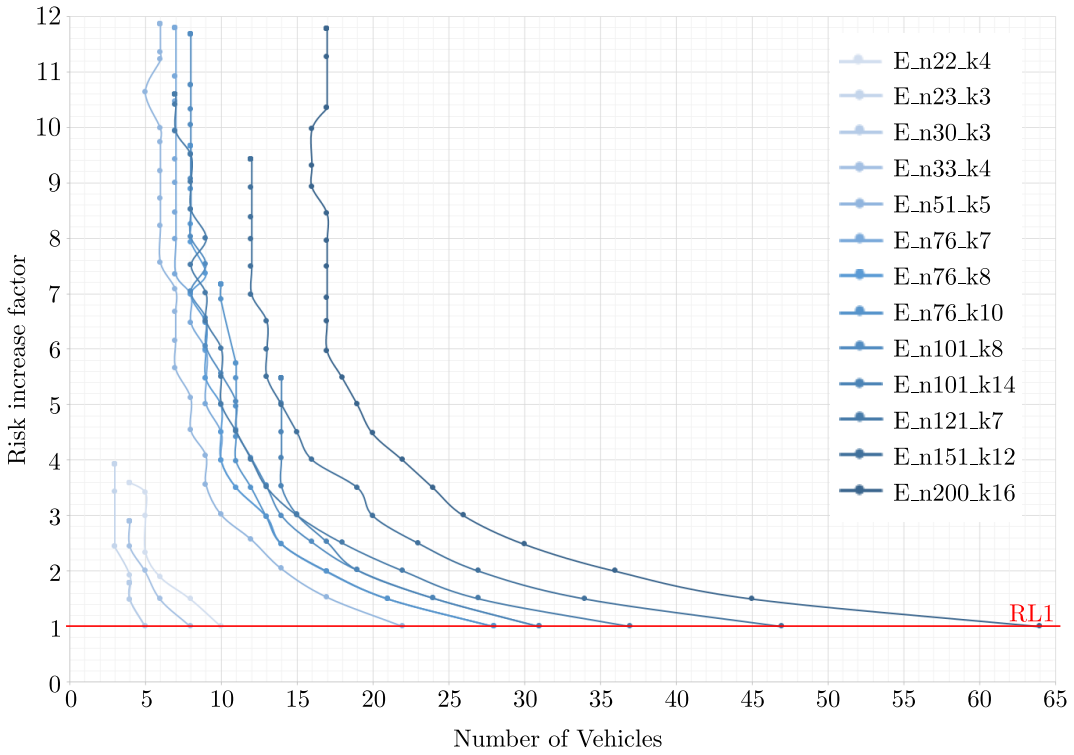


FIGURE 6.4: *The anticipated risk increase associated with the number of vehicles used for the maximum anticipated risk on a single route for each of the thirteen problem instances.*

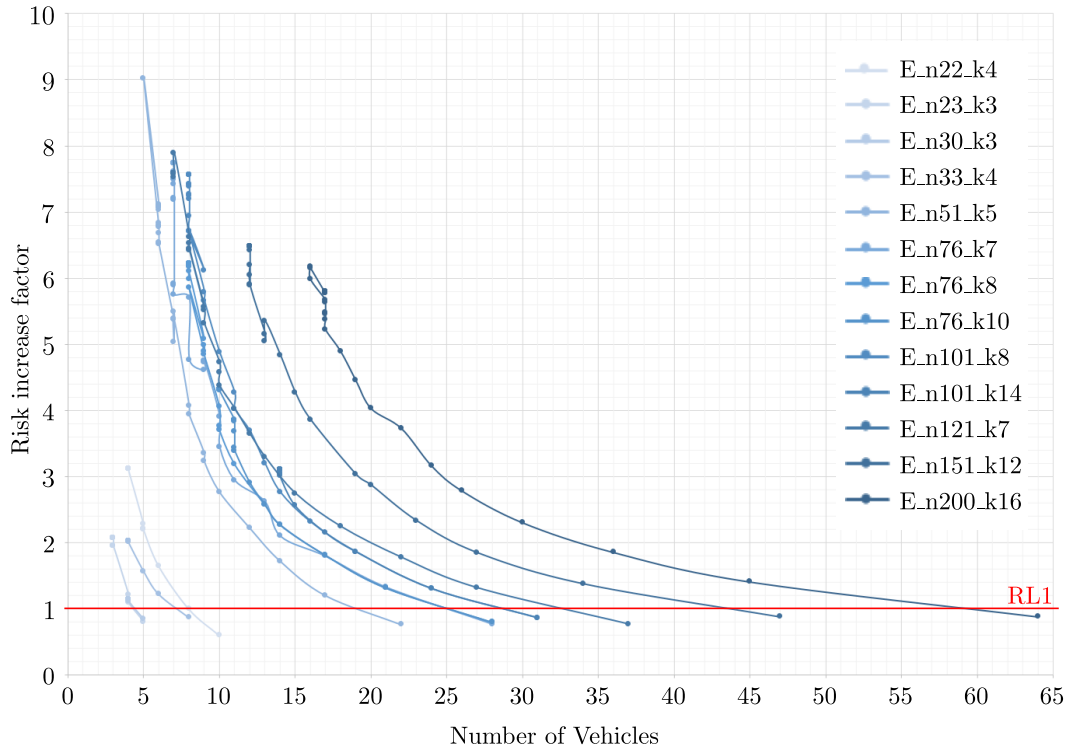


FIGURE 6.5: *The anticipated risk increase associated with the number of vehicles used for the average anticipated risk on a route for each of the thirteen problem instances.*

The x-axis denotes the number of vehicles required to solve the problem without violating any constraints, while the y-axis denotes the risk increase factor that may be expected if the risk threshold level is made more lenient. The red line indicates the minimum allowable risk threshold (RL1) for each problem instance (*i.e.* risk increase factor $=$ 1). It may be observed that all the data points under the red line are the cases where the risk of the routes are strictly constrained and, therefore the average risk increase factor for the iteration is less than 1. This means that the route with the maximum possible risk will be equal to $RL1$ (as shown previously in Figure 6.4) for a problem instance with a risk threshold of $RL1$. For example, when analysing E_n200_k16 it may be observed that when the risk threshold is at its strictest (*i.e.* equal to $RL1$) the solution requires 64 vehicles and the *average risk* on the route will be equal to a factor 0.8 or the average risk will reduce by a factor equal to $1 - 0.8 = 0.2$ (*i.e.* 0.2 less than the minimum allowable risk threshold), however, when the risk threshold is at the most lenient (*i.e.* equal to $RL15$) the solution requires seventeen vehicles and the average risk on the route will increase by a factor 5.8 (more than the minimum allowable risk threshold). For completeness, the data used for the graph in Figure 6.4 and 6.5 is illustrated in Appendix A.

Next, the risk associated with each of the problem instances was normalised in order to observe if the same effect is observed in each of the problem instances (*i.e.* that more vehicles are used for a problem with a stricter risk threshold and that less vehicles are used for problems with a more lenient risk threshold). The normalised risk value provides a value between zero and one and is calculated as

$$\text{Normalised value} = \frac{R - R_{\min}}{R_{\max} - R_{\min}}, \tag{6.6}$$

where the risk of the current route is denoted by R, the route with the least anticipated risk in the solution is denoted by $R_{\min}$ and the route with the most anticipated risk in the solution is denoted by $R_{\max}$.

The normalised results for each of the thirteen problem instances listed in Table 6.3 combined with the twenty nine different risk levels are illustrated graphically in Figure 6.6.

The x-axis denotes the number of vehicles required to solve the problem without violating any of the constraints, while the y-axis denotes the normalised risk associated with each problem instance. Figure 6.6 provides an alternative illustration of the anticipated risk that is associated with the number of vehicles used to solve the problem. It is easy to see that as the number of vehicles used to solve the problem decreases, the risk associated with the problem solution increases. This is anticipated, because if less vehicles are used, they will be required to carry more valuable goods and travel longer distances, which, in turn, increases the risk accumulation along a route. If more vehicles are used, however, they will not be required to carry as many valuable goods individually and they will travel shorter distances individually, which, in turn, accumulates less risk along each route. From Figure 6.6, it may be observed that a normalised risk $=$ 0 illustrates the strictest risk threshold ($RL1$) which, in an ideal world, would incur no risk. A normalised risk $=$ 1 illustrates the most lenient risk threshold ($RL15$) which gives an indication of the risk associated with the worst case scenario, if a heist were to occur.

From the data and calculations represented in Table 6.3 and in Figures 6.4 and 6.6, it may be concluded that the model is valid and computationally robust. These data and graphs confirm that the RCTVRP is able to produce vehicle routes that are subject to risk constraints and that these routes are valid. The routes changes as the risk threshold enforced on the problem is made more lenient and, therefore it may be concluded that the RCTVRP model does work correctly and as was anticipated. Furthermore, the model output serves as a validation of the expected outcome, rather than a prediction of a certain outcome.

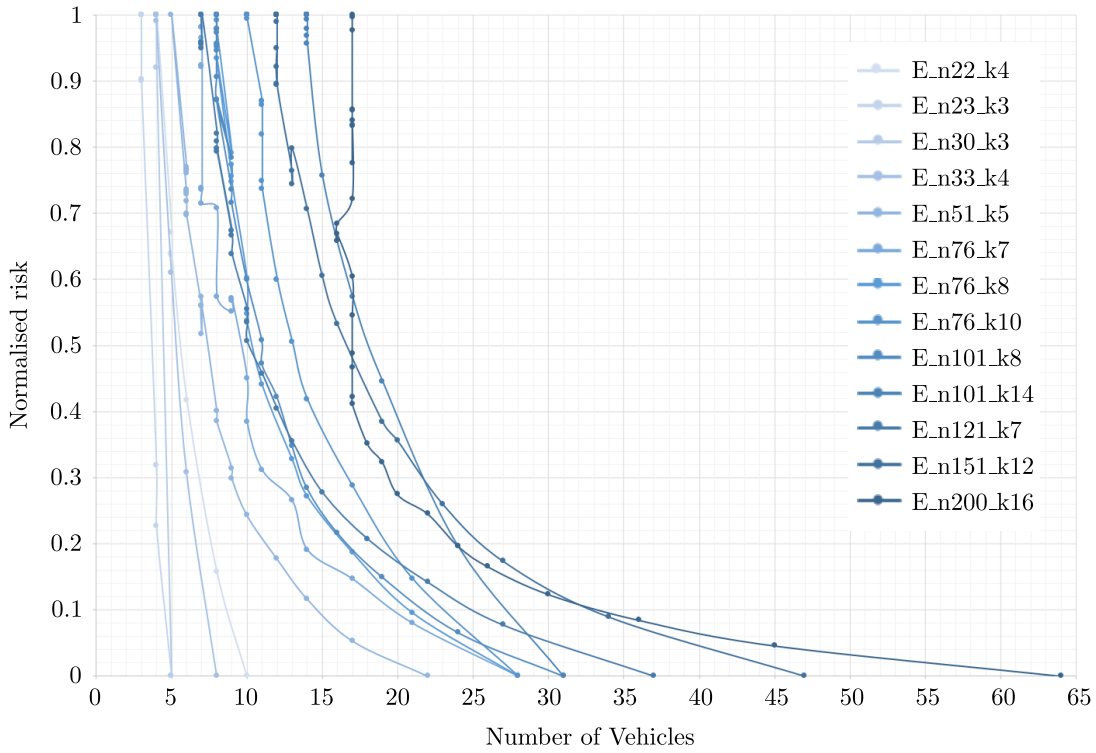


FIGURE 6.6: *The normalised risk of each problem instance versus the number of vehicles required to solve the problem without violating any constraints.*

For the purpose of completeness, the risk increase factors for the different problem instances were also calculated without the influence of a capacity constraint (*i.e.* only the risk constraint is enforced), in order to observe if the trend persists. The risk increase factor results for the *maximum risk on a single route* for each of the thirteen problem instances listed in Table 6.3 combined with the twenty nine different risk levels and without capacity constraints are illustrated graphically in Figure 6.7.

In a similar fashion as was described previously, the x-axis denotes the number of vehicles required to solve the problem without violating any constraints, while the y-axis denotes the risk increase factor that may be expected if the risk threshold level is made more lenient. The red line indicates the minimum allowable risk threshold ($RL1$) for each problem instance (*i.e.* risk increase factor = 1). It may be observed that all the data points on the red line are the cases where the risk of the routes are the most strictly constrained and, therefore the risk increase factor is equal to 1. For example, when analysing E_n200_k16 it may be observed that when the risk threshold is at its strictest (*i.e.* equal to $RL1$) the solution requires 64 vehicles and the route with the maximum anticipated risk will be equal to a factor one (*i.e.* equal to the minimum allowable risk threshold), which is the same as in the capacitated case in Figure 6.4. When the risk threshold is the most lenient (*i.e.* equal to $RL15$) the solution only requires ten vehicles, however, the maximum anticipated risk will increase by a factor 14.9, which is much higher than the capacitated case in Figure 6.4.

Similarly, the risk increase results for the *average risk for the fleet of vehicles* used for each of the thirteen problem instances listed in Table 6.3 combined with the twenty nine different risk levels and without capacity constraints are illustrated graphically in Figure 6.8.

The x-axis denotes the number of vehicles required to solve the problem without violating any constraints, while the y-axis denotes the risk increase factor that may be expected if the risk

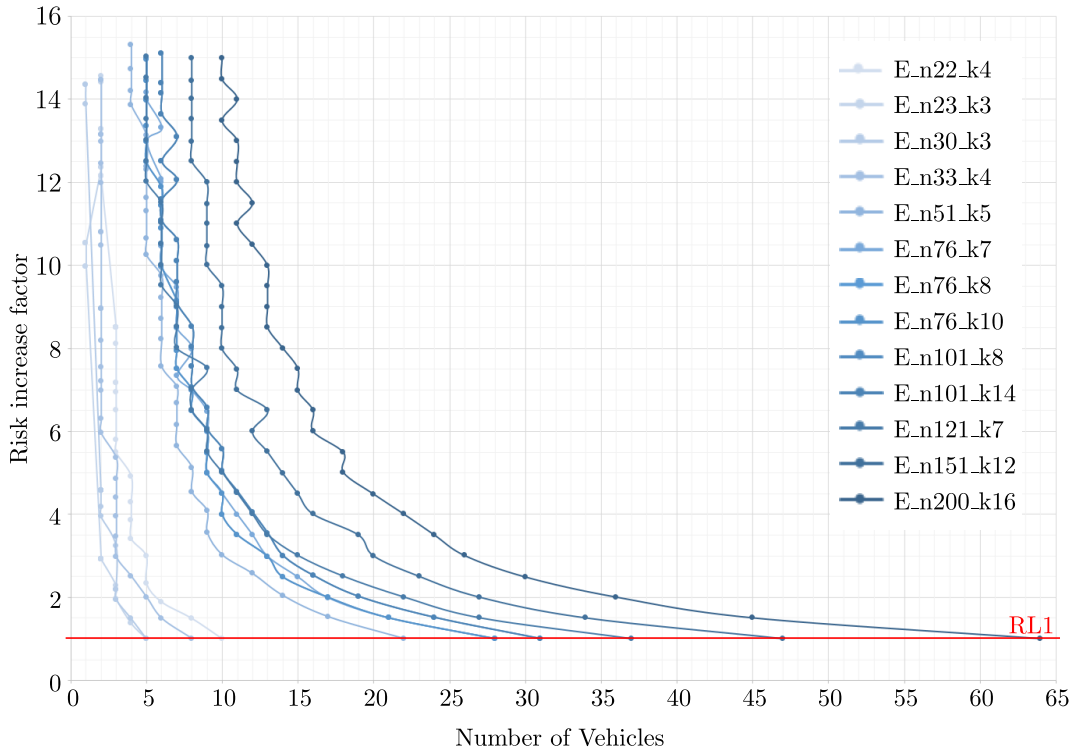


FIGURE 6.7: *The anticipated risk increase associated with the number of vehicles used for the maximum anticipated risk on a single route for each of the thirteen problem instances, without capacity constraints.*

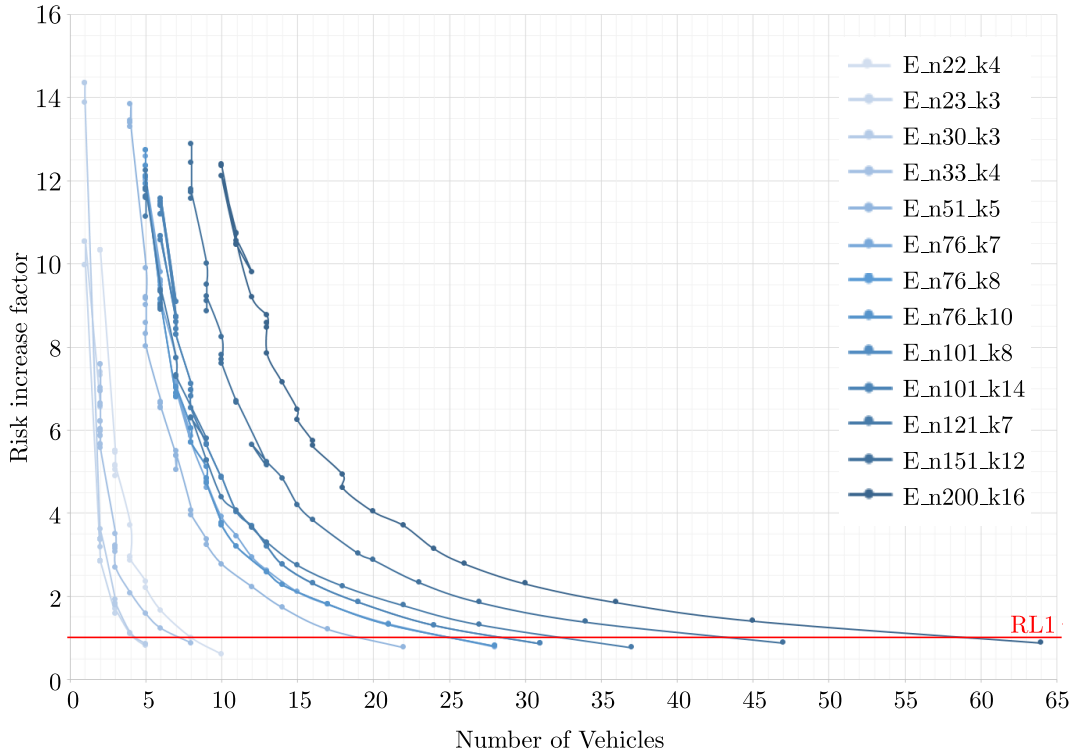


FIGURE 6.8: *The anticipated risk increase associated with the number of vehicles used for the average anticipated risk on a route for each of the thirteen problem instances, without capacity constraints.*

threshold level is made more lenient. The red line indicates the minimum allowable risk threshold (RL1) for each problem instance (*i.e.* risk increase factor = 1). It may be observed that all the data points under the red line are the cases where the risk of the routes are strictly constrained and, therefore the average risk increase factor for the iteration is less than 1. This means that the route with the maximum possible risk will be equal to $RL1$ (as shown previously in Figure 6.4) for a problem instance with a risk threshold of $RL1$. For example, when analysing E_n200_k16 it may be observed that when the risk threshold is at its strictest (*i.e.* equal to $RL1$) the solution requires 64 vehicles and the *average risk* on the route will be equal to a factor 0.8 or the average risk will reduce by a factor equal to $1 - 0.8 = 0.2$ (*i.e.* 0.2 less than the minimum allowable risk threshold), which is the same as in the capacitated case in Figure 6.4. When the risk threshold is at the most lenient (*i.e.* equal to $RL15$) the solution requires ten vehicles, however, the average risk on the route will increase by a factor 12.4, which is much higher than the capacitated case in Figure 6.4. For completeness, the data used for the graph in Figure 6.7 and 6.8 is illustrated in Appendix B.

In conclusion, both variations of the sensitivity analyses therefore improve the confidence level in the developed model since the model succeeded in providing similar results as was expected from the theoretical benchmark solutions. Since the variation from the achieved results to the target results are comparatively small, the variation may be considered as acceptable in the context of this particular application and the model may therefore be regarded as valid. Furthermore, it may be observed that as the risk threshold is made more lenient, there are less vehicles required to solve the problem, however, the anticipated risk will increase. The anticipated risk will increase even more if there is no capacity constraint enforced on the vehicle. From the validation and results, it may, therefore, be derived that it is better to enforce a capacity constraint along with a risk threshold in order to achieve an improvement in mitigating risk along a route.

6.3 Chapter summary

In this chapter, the model along with the two implemented algorithms (CVRP and RCTVRP) were validated according to the methods discussed in Chapter 4.

Section 6.1 contained a detailed description of the methods that were used to verify whether the model was built according to the user-specified requirements. These methods included a top-down and modular development approach, accompanied by a variety of "checking" mechanisms. Furthermore, section §6.2 contained a detailed description of the methods that were used to validate the model. The first method was a trace validation which was discussed along with examples in section §6.2.1. The second method of validation was a sensitivity analysis which was discussed in section §6.2.2. The sensitivity analysis was conducted in two parts. The first part of the sensitivity analysis aimed to determine the computational robustness of the CVRP algorithm and was tested using test data from ten benchmark problems available in the literature. It was found that the developed model provides good results when compared to the results of the ten benchmark problems. The second part of the sensitivity analysis aimed to ascertain how a change in some parameters impact the performance and results provided by the RCTVRP algorithm. The risk threshold parameter was varied between an upper and a lower bound to see how the model performs and what the solutions look like. It was observed that if the risk threshold is made strict, the model required more vehicles to serve the customers, while if the risk threshold is made more lenient, the model required less vehicles to serve the customers. The results were also illustrated graphically in order for the user to observe the trend. The chapter finally closed with a brief summary on the chapter contents.

Part III

Conclusion

CHAPTER 7

Conclusion

Contents

This thesis conclusion chapter comprises 3 sections. A summary of the contents of the thesis is provided in a chapter-by-chapter overview in §7.1, while the contributions of the thesis are listed and discussed in §7.2. Finally, the chapter closes in §7.3 with suggestions for possible avenues for future work or further investigation that builds on the contributions of this thesis. The future work is suggested in two primary categories, where the first category is in relation to the modelling, formulation and solution approaches of the RCTVRP model, while the second category is in relation to the simulation modelling that may be incorporated for the RCTVRP.

7.1 Thesis summary

Apart from this concluding chapter, this thesis contains six chapters. Each of these chapters are briefly summarised in this section.

Chapter 1 contained a brief background on CIT vehicle operating procedures and how these vehicles are used to transport valuable goods. The chapter provided some statistics on the relevance of cash circulation in South Africa, as well as in a global context. A discussion, which highlighted the risks involved in the transportation of valuable goods followed in fulfilment of Objective I (a) of §1.3. A brief overview was also provided of other approaches that have been pursued to combat risk in the context of CIT vehicle routing. A discussion followed on the thesis problem statement, the seven objectives that were pursued in the thesis, the thesis scope and the research methodology that was followed throughout the project.

A comprehensive literature review was provided in Chapter 2 of the operations research literature pertaining to the VRP and its solution approaches in fulfilment of Objectives I (c), (d), (f) and (g) of §1.3. The review covered literature on different formulations of the VRP, including the travelling salesman problem, the capacity constrained VRP, the distance constrained VRP, the time window constrained VRP, the precedence constrained VRP, the risk constrained VRP and

the dynamic VRP. Furthermore, typical solution approaches that may be used to effectively solve for variants of the VRP were discussed and included the Clarke-Wright savings algorithm, the simulated annealing algorithm, the tabu search algorithm, the genetic search algorithm and the ant colony optimisation algorithm.

Furthermore, in Chapter 3 a brief review of the operations research literature pertaining to simulation modelling is provided in fulfilment of Objective I (b) of §1.3. The review covered literature on the history of simulation and the evolution thereof. The review also included a discussion on the various simulation paradigms which include static versus dynamic models, deterministic versus stochastic models and continuous versus discrete models. In addition, the different levels of abstraction (*i.e.* low, medium and high abstraction) involved in the development of a simulation model were highlighted, as well as which level of abstraction is typically related to the different types of simulation models, including discrete event simulation modelling, agent-based simulation modelling, system dynamics simulation modelling and dynamic systems simulation modelling. The review described the different elements involved in a simulation model and the advantages and disadvantages associated with simulation modelling. Finally, the steps involved in the development of a simulation model were discussed in order to provide a sufficient understanding for the development of a DSS towards a simulation model and, lastly, there was a discussion on the difference between optimisation and simulation.

In Chapter 4, DSS frameworks in general from the literature were discussed in fulfilment of Objective I (e) and (h) of §1.3. The review covered literature on frameworks with a specific focus on decision support frameworks, the structure, the integrated components and system software that are encompassed in a DSS. Typical systems development methodology approaches including the waterfall methodology, the agile methodology and the object-oriented methodology were also discussed. Furthermore, research was conducted on the notion of testing, training and maintaining the system, which included discussions on quality assurance, various testing methods, training techniques, system implementation and system maintenance methods.

Chapter 5 was devoted to a description of the developed DSS framework developed in fulfilment of Objective II–IV and VI of §1.3. First, the modelling software that was used to develop the DSS was discussed along with its features. The implemented DSS contains three main components namely the database, the user interface and the model base. The model base is responsible for housing two models namely the CVRP model and the RCTVRP model. A user is allowed to provide specific input and parameter information through the GUI and this information is then considered, processed and optimised according to the user's preferences. The results displayed by the DSS are displayed to the user again through a GUI, where the model output serves as a validation of the expected outcome. The chapter provided the algorithm frameworks for each of the two models used in the model base. Furthermore, the implementation in the modelling environment was discussed by first defining the object classes involved in the model. Next, the GUI was described in the context of the modelling environment, as well as how the user is able to initialise the model through the use of the GUI. Finally, the model optimisation and visualisation were discussed in the context of the modelling environment.

Chapter 6 was dedicated to testing and validating the working of the DSS proposed in Chapter 5 in fulfilment of Objective V of §1.3. First, the methods used to verify the working of the model were described in order to confirm that the model does indeed work according to the user-specified requirements. Furthermore, the model was validated using two techniques that were discussed in Chapter 4 including trace validation and a sensitivity analysis, in order to confirm that the model works correctly. The sensitivity analysis was performed in two parts, where the first part uses test data from the benchmark VRP library to validate the working of

the CVRP model. The second part was a parameter variation that was implemented on the RCTVRP model in order to validate that it produces the correct results.

7.2 The contributions of this study

Six main contributions were made in this thesis. These contributions are discussed in this section.

Contribution 1 *An extensive review of the literature related to VRPs and some of the solution approaches that may be used to solve VRPs.*

In Chapter 2, an extensive review was given of the history and evolution of the VRP since its first formulation in 1959 (in fulfilment of Objective I (a) of §1.3). The different VRPs are presented since its inception in 1959 and the review spanned the entire time period up to the present. The description of VRPs was divided into two main sections, where the first section focused on different VRP formulations and the second section focused on VRP solution approaches.

Contribution 2 *The proposed risk mitigating DSS tool for investigating different scenarios which may be used towards the development of a simulation model that may be used as a stepping-stone to build a fully functional simulation model.*

The major contribution of this thesis is the risk mitigating DSS tool developed in Chapter 5 (in fulfilment of Objectives II, III and VI of §1.3). As was mentioned in §1.2, the main aim of this thesis is to develop a DSS framework that may be used towards the development of a simulation model which is capable of suggesting suitable vehicle routes in order to mitigate risk within the CIT industry. The purpose of the DSS is to provide the user with a reference of how the capacity load on a vehicle and the distance travelled by a vehicle affects the anticipated risk along a route and also give an indication of the number of vehicles that should be used to service the customers.

Contribution 3 *The formulation of an interactive DSS model that is configurable by the user and is able to optimise and provide insights with respect to risk mitigation along routes and present the routes visually through a GUI.*

The developed DSS of Chapter 5 was developed to be interactive with the user. The user is able to specify his or her requirements by configuring the model constraints and parameters through the use of a GUI (in fulfilment of Objective IV of §1.3). The GUI is equipped with various features, such as sliders, tick boxes and text boxes that makes the configuration process even more user-friendly. These features where implemented along the notion of a *settings* tab which contains the most notable constraints and parameters. The parameters and constraint may be configured by the user to model different problem scenarios (in fulfilment of Objective VI of §1.3). Once the model is configured and executed, the user is presented with another GUI which again accommodates different features and informative chart displays which may be evaluated by the user in order to draw certain conclusions. Furthermore, the animation features available in the modelling environment were used to visually represent the customers, the depot, the vehicles and the various routes each vehicle follows.

At this point it is important to also highlight that the DSS model developed in this thesis is not meant to replace any existing models, however, it is envisioned that it may be used to gain insight from another perspective of viewing and interpreting the anticipated risk that may by encountered on a vehicle route.

Contribution 4 *The implementation of the modified Clarke-Wright savings algorithm in the context of a RCTVRP in the modelling environment.*

The modified Clarke-Wright savings algorithm was formulated by Talarico *et al.* [126] in 2015 and is based on a number of assumptions and insights that were obtained from the literature and aims to produce suitable vehicle routes, while mitigating risk for CIT vehicles. Since the modelling environment used in this thesis is not a natural optimisation environment and was used with the purpose of creating opportunities for simulation modelling, the algorithm implementation in the modelling environment proved to be quite complex. The implementation of the algorithm in the modelling environment was successful despite the challenges and is well documented in Chapter 5 and validated in Chapter 6 (and works in fulfilment of Objective III of §1.3).

Contribution 5 *A sensitivity analysis in respect of the effect of a varying risk threshold on the number of vehicles and the distance travelled by these vehicles.*

A sensitivity analysis was performed in respect of the effect of a varying risk threshold on the number of vehicles and the distance travelled by these vehicles in §6.2.2 and is in fulfilment of Objective V of §1.3. The analysis illustrated the effects of different risk thresholds on the number of vehicles used to solve a problem, and it was observed that stricter risk thresholds on vehicle routes required more vehicles in order to solve the RCTVRP, while more lenient risk thresholds allowed for less vehicles to be required in order to solve the RCTVRP.

Contribution 6 *The suggestion and prioritisation of ideas that may be considered for future work following on the contributions of this thesis.*

The final contribution of this thesis is discussed in the next section. The suggestions listed in the next section are made in an effort to help orientate the development of RCTVRP models in the future by highlighting and discussing suitable avenues of investigation as possible future work that builds on the contributions already provided in this thesis (in fulfilment of Objective VII of §1.3).

7.3 Possible future work

A number of suggestions for possible future work with respect to the RCTVRP DSS model, which transpired during the development of the DSS model and during the compilation of this thesis, are presented in this section. Two primary categories of suggestions emerged, where the first category is in relation to the modelling, formulation and solution approaches of the RCTVRP model, while the second category is in relation to the simulation modelling that may be incorporated for the RCTVRP model.

7.3.1 Modelling, formulation and solution approaches of the RCTVRP model

The first and most prominent suggestion is to enlarge the scope of the thesis. It would be beneficial to include demand collections and deliveries from customers, because when vehicles are allowed to deliver valuable goods along its route, the risk is lowered (due to less capacity on board the vehicle) and, therefore, vehicles may be able to travel slightly longer routes and serve more customers and subsequently require less vehicles. Furthermore, it was initially envisioned that dynamic demand would be a realistic incorporation, however, in reality CIT vehicle routes are planned on the day of execution and are not changed dynamically for security reasons. It may, however, be beneficial to incorporate time windows to some extent, since customers typically

expect deliveries or collections to happen at a set time. Furthermore, it may be beneficial to explore collaboration between the model presented by Talarico *et al.* [126] and the model presented by Yan *et al.* [150] as discussed in Chapter 2. This collaboration would then typically include the risk threshold constraint proposed by Talarico *et al.* [126], while also considering time and space as is proposed by the model of Yan *et al.* [150].

The scope may be enlarged even further by incorporating a non-homogeneous vehicle fleet. In the CIT industry, vehicles are rated according to how secure they are. More secure vehicles may carry more valuable goods (*i.e.* a higher capacity at a better insurance rate), while less secure vehicles are expected to carry less valuable goods (*i.e.* a lower capacity at a higher insurance rate).

Various solution approaches were discussed in Chapter 2, however, the solution approach taken for this thesis was the Clarke-Wright savings algorithm, as well as the RCTVRP modified version thereof. It may be beneficial to explore some of the other heuristics and meta heuristics to observe if they perform better or the same in the context of the RCTVRP.

Finally, this particular study was limited greatly due to the lack of real-world case study data. It is therefore suggested that future endeavours are partnered with industry partners or stakeholders that are able to supply the researcher with realistic and comprehensive data. The study of real-world data is beneficial, since it provides a more clear anticipation of the impacts of risk on CIT vehicle routes.

7.3.2 Simulation modelling that may be incorporated for the RCTVRP model

The AnyLogic modelling software used in this thesis provides many features that may be beneficial in the analysis of risk on routes in a simulation context. Due to time and real-world data constraints that were experienced during the execution process of this thesis, however, the researcher did not implement simulation modelling. The researcher chose to focus on the optimisation of the model and focus on developing a validation model, rather than developing a predictive model using simulation (based on Chapter 3) due to the time constraints. The developed DSS is, however, developed and implemented in an environment that welcomes the addition of simulation in possible future work. The accurate development and implementation of simulation modelling is, however, very costly and time consuming, however, simulation modelling does provide many benefits as were described in Chapter 3 and, therefore, some future work with the focus on simulation methods are suggested — if the user and developer are willing to sacrifice sufficient time and resources to further explore the suggestions provided in this section.

Firstly, the incorporation of GIS locations in terms of realistic routes along existing roads and accurate customer locations using GPS coordinates may make the problem appear more realistic. Furthermore, the inclusion of GIS locations could be beneficial in highlighting so-called "criminal hot spot" areas in order for vehicles to avoid routes that lead through these areas as much as possible. These "criminal hot spot" areas may be identified by analysing historical data related to CIT heists, as well as which areas are generally more prone to high-jacking situations and other criminal activities.

Secondly, research may be done on the historical events of CIT heists and an impact analysis may be carried out in order to quantify the impact of such a heist. Typical CIT heists may then be simulated according to this past historical data in order to attempt to quantify the anticipated impact of a particular heist scenario. The factors involved in a CIT heist may be analysed and translated as simulation features that attempt to accurately represent these factors. The inclusion of such factors and features may lead to an even more realistic model by building its

working and analysis on actual past data. The addition of simulation modelling may enable the model to provide a predicted set of results which may enable the user to anticipate and analyse certain impacts before they occur. Predictive results may be able to assist the user in understanding what the likelihood is that unwanted events may occur and what the anticipated cost impact may be in the event that they do occur. The addition and implementation of such a feature is very time consuming and costly, therefore, the current model may be used as a basis to decide how much time or resources are worth investing in the exploration of a more complex model in order to effectively mitigate risk.

Thirdly, it is commonly known that in many CIT heists, the robbers involved in the heist have accomplices in the vehicles. Agent-based simulation modelling (as discussed in Chapter 3) may be able to address this issue to some extent. Agent-based modelling approaches are able to assign certain features, characteristics and behavioural traits to many different individual agents in a system. Even though this particular addition of the simulation modelling features may be purely speculative, it may be able to provide insights into the impacts associated with CIT heists where there are accomplices involved.

Lastly, in the context of real-world scenarios, simulation models are typically able to represent reality in a relatively close representation. If a simulation model may be constructed to incorporate GIS capabilities, it is possible to update the model in real-time by analysing roads, road closures, traffic and other events that might disrupt routing during the day. This provides the user with some measure to combat uncertainty that may arise during a typical day and would enable the user to make real-time decisions and observe the anticipated impact thereof before the decisions are actually implemented or executed in real-life. The visual features that are available in the ANYLOGIC modelling software, make this addition appealing since the user will be able to visually observe and track the vehicles in a real-time manner.

References

[1] Abar S, Theodoropoulos GK, Lemarinier P & O'Hare GMP, 2017, *Agent-based modelling and simulation tools: A review of the state-of-art software*, Computer Science Review, **24**, pp. 13–33.

[2] Adendorff SA & Kruger PS, 2011, *A decision support model for the cash replenishment process in South African retail banking*, The South African Journal of Industrial Engineering, **11(2)**, pp. 1–26.

[3] Aloise D, Duhamel C & Santos A, 2008, *ModelLing the mobile oil recovery problem as a multi-objective vehicle routing problem*, Proceedings of the second International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences, Berlin, pp. 283–292.

[4] Altinkemer K & Gavish B, 1991, *Parallel savings-based heuristics for the delivery problem*, Operations Research, **39(3)**, pp. 456–469.

[5] Altinel IK & Öncan T, 2005, *A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem*, Journal of the Operational Research Society, **56(8)**, pp. 954–961.

[6] AnyLogic, 1999, *AnyLogic*, [Online], [Cited March 2017], Available from `http://www.anylogic.com`.

[7] Baldacci R, Toth P & Vigo D, 2007, *Recent advances in vehicle routing exact algorithms*, 4OR, **5(4)**, pp. 269–298.

[8] Banks J, 2000, *Introduction to simulation*, Proceedings of the first Simulation Conference, Oxford, pp. 9–16.

[9] Banks J, 1998, *Principles of simulation*, pp. 3–30 in Banks J (Ed), *Handbook of simulation: Principles, methodology, advances, applications, and practice*, John Wiley & Sons, Canada (CA).

[10] Bekker J, 2016, *Steps in a simulation study*, pp. 24–25 in Bekker J (Ed), *Simulation 442, Lecture Notes*, Stellenbosch University, Stellenbosch.

[11] Bertsimas DJ & Simchi-Levi D, 1996, *A new generation of vehicle routing research: robust algorithms, addressing uncertainty*, INFORMS Journal on Operations Research, **44(2)**, pp. 286–304.

[12] Bertsimas DJ & Van Ryzin G, 1991, *A stochastic and dynamic vehicle routing problem in the Euclidean plane*, INFORMS Journal on Operations Research, **39(4)**, pp. 601–615.

[13] Bertsimas DJ & Van Ryzin G, 1993, *Stochastic and dynamic vehicle routing in the Euclidean plane with multiple capacitated vehicles*, INFORMS Journal on Operations Research, **41(1)**, pp. 60–76.

[14] Bertsimas D & Tsitsiklis J, 1993, *Simulated annealing*, Statistical science, **8(1)**, pp. 10–15.

[15] Bianco L, Caramia M, Giordani S & Piccialli V, 2013, *Operations research models for global route planning in hazardous material transportation*, pp. 49–101 in Batta R & Kwon C (Eds), *Handbook of OR/MS Models in Hazardous Materials Transportation*, Springer, Chile.

[16] Bierlaire M, 2015, *Simulation and optimization: A short review*, Transportation Research Part C: Emerging Technologies, **55**, pp. 4–13.

[17] Blum C, 2005, *Ant colony optimization: Introduction and recent trends*, Physics of Life Reviews, **2(4)**, pp. 353–373.

[18] Bolton D, 2018, *C++ for beginners*, [Online], [Cited August 2018], Available from `https://www.thoughtco.com/candand-for-beginners-958278`.

[19] Bredstrom D & Ronnqvist M, 2008, *Combined vehicle routing and scheduling with temporal precedence and synchronization constraints*, European Journal of Operational Research, **191(1)**, pp. 19–31.

[20] British Security Industry Association, 2018, *Combating cash crime*, [Online], [Cited April 2018], Available from `https://www.bsia.co.uk/cvicrime#`.

[21] Broshchev A & Fililppov A, 2004, *From system dynamics and discrete event to practical agent-based modelling: Reasons, techniques, tools*, Proceedings of the twenty second International conference of the system dynamics society, Oxford.

[22] Bullnheimer B, Hartl RF & Strauss C, 1999, *Applying the ant system to the vehicle routing problem*, pp. 285–296 in Voss S, Martello S, Osman IH & Roucairol C (Eds), *Meta-heuristics*, Springer, Vienna.

[23] Busetti F, 2003, *Simulated annealing overview*, [Online], [Cited November 2008], Available from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.5018&rep=rep1&type=pdf`.

[24] Business Dictionary, 2018, *Definition of Risk*, [Online], [Cited April 2018], Available from `http://www.businessdictionary.com/definition/risk.html`.

[25] Business Live, 2016, *Declining client fees threaten cash-in-transit industry*, [Online], [Cited July 2018], Available from `https://www.businesslive.co.za/bd/companies/trade-and-industry/2016-07-13-declining-client-fees-threaten-cash-in-transit-industry/`.

[26] Carson Y & Maria A, 1997, *Simulation optimization: Methods and applications*, Proceedings of the twenty-ninth Winter simulation conference, New York, pp. 118–126.

[27] Center for Chemical Process Safety, 2008, *Guidelines for Chemical Transportation Safety, Security and Risk Management*, John Wiley & Sons, New York (NY).

[28] Chiang WC & Russell RA, 1996, *Simulated annealing metaheuristics for the vehicle routing problem with time windows*, Annals of Operations Research, **63(1)**, pp. 3–27.

[29] Clarke G & Wright JW, 1964, *Scheduling of vehicles from a central depot to a number of delivery points*, INFROMS Journal on Operations research, **12(4)**, pp. 568–581.

[30] Cordeau JF, Gendreau M, Hertz A, Laporte G & Sormany JS, 2005, *New heuristics for the vehicle routing problem*, pp. 279–297 in Langevin A & Riopel D (Eds), *Logistics systems: Design and optimization*, Springer, Canada (CA).

[31] Cordeau JF, Gendreau M, Laporte G, Potvin JY & Semet F, 2002, *A guide to vehicle routing heuristics*, Journal of Numerical Analysis, Industrial and Applied Mathematics, **39(3)**, pp. 456–469.

[32] Cordeau JF, Laporte G, Savelsbergh MWP & Vigo D, 2007, *Vehicle Routing*, Handbook in operations research and management science, **14**, pp. 367–428.

[33] Coyle T, 2016, *How to train employees on a new software rollout*, [Online], [Cited October 2018], Available from `https://trainingindustry.com/articles/learning-technologies/how-to-train-employees-on-a-new-software-rollout/`.

[34] Dantzig GB & Ramser JH, 1959, *The truck dispatching problem*, Management Science, **6(1)**, pp. 80–91.

[35] David B, 2018, *Procedure in training employees*, [Online], [Cited September 2018], Available from `https://smallbusiness.chron.com/procedures-training-employees-42799.html`.

[36] Davis L & Williams G, 1994, *Evaluating and selecting simulation software using the analytic hierarchy process*, Integrated Manufacturing Systems, **5(1)**, pp. 23–32.

[37] Deneubourg JL, Aron S, Goss S & Pasteels JM, 1990, *The self-organizing exploratory pattern of the argentine ant*, Journal of Insect Behavior, **3(2)**, pp. 159–168.

[38] Dennis A, Wixom B & Tegarden D, 2005, *Systems Analysis and Design with UML Version 2.0: An Object Oriented Approach*, 2nd Edition, John Wiley & Sons, Hoboken (NJ).

[39] Deogun JS, 1988, *A conceptual approach to decision support system models*, Information Processing & Management, **24(4)**, pp. 429–448.

[40] Dessouky M, Ordonez F & Sungur I, 2007, *A priori performance measures for arc-based formulations of the vehicle routing problem*, Transportation Research Record, **2032**, pp. 53–62.

[41] Ding Q, Hu X, Sun L & Wang Y, 2012, *An improved ant colony optimization and it's application to vehicle routing problem with time windows*, Neurocomputing, **98**, pp. 101–107.

[42] Dorigo M, Birattari M & Stutzle T, 2006, *Ant colony optimization*, IEEE Computational Intelligence Magazine, **1(4)**, pp. 28–39.

[43] Dorigo M, Maniezzo V & Colorni A, 1992, *Distributed optimization by ant colonies*, Proceedings of the first European Conference on Artificial Life, Paris, pp. 134–142.

[44] Doyuran T & Çatay B, 2011, *A robust enhancement to the Clarke–Wright savings algorithm*, Journal of the Operational Research Society, **62(1)**, pp. 223–231.

[45] Dréo J, Pétrowski A, Siarry P & Taillard E, 2006, *Metaheuristics for hard optimization: Methods and case studiess*, Springer, New York (NY).

[46] Elite Data Science, 2018, *Data cleaning*, [Online], [Cited October 2018], Available from `https://elitedatascience.com/data-cleaning`.

[47] Erasmus A, 2011, *The cash in transit industry in South Africa: It's challenges and solutions*, Presentation, Protea Coin Group.

[48] Fandom, 2018, *Hierarchical Data Model*, [Online], [Cited March 2018], Available from `http://databasemanagement.wikia.com/wiki/Category:Hierarchical_Data_Model`.

[49] Francq P, 2012, *Optimisation problems*, [Online], [Cited January 2018], Available from `http://www.otlet-institute.org/wikics/Optimization_Problems.html`.

[50] French CD, 1995, *One size fits all database architectures do not work for DSS*, Proceedings of the second Association for Computing Machinery SIGMOD Record, San Jose, pp. 449–450.

[51] Galin D, 2004, *Software quality factors*, pp. 36–54 in Mansfield K (Ed), *Software quality assurance: From theory to implementation*, Pearson Education India, England.

[52] Gambardella L, Taillard E & Agazzi G, 1999, *MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows*, pp. 1–17 in Corne D, Dorigo M & Glover F (Eds), *New Ideas in Optimization*, McGraw-Hill, Lugano.

[53] Gaskell T, 1967, *Bases for vehicle fleet scheduling*, Journal of the Operational Research Society, **18(3)**, pp. 281–295.

[54] Goldsman D, Nance RE & Wilson JR, 2010, *A brief history of simulation revisited*, Proceedings of the winter simulation conference, Baltimore, pp. 567–574.

[55] GoodReads, 2018, *Albert Einstein: Quotes*, [Online], [Cited October 2018], Available from `https://www.goodreads.com/author/quotes/9810.Albert_Einstein`.

[56] Grigotyev I, 2015, *AnyLogic 7 in three days: A quick course in simulation*, AnyLogic.

[57] Harrell C, Gosh B & Bowden R, 2011, *Simulation Basics*, pp. 57–85 in McGraw-Hill (Ed), *Simulation using ProModel*, McGraw-Hill Education, Boston (MA).

[58] Heylighen F, 2016, *Stigmergy as a universal coordination mechanism II: Varieties and evolution*, Cognitive Systems Research, **38**, pp. 50–59.

[59] Hillier FS, Lieberman GJ, Nag B & Basu P, 2005, *Introduction to Operations Research*, Hash DB (Ed), *Introduction to Operations Research*, McGraw-Hill, New York (NY).

[60] Hoffman KL, Padberg M & Rinaldi G, 2013, *Traveling salesman problem*, pp. 1573–1578 in Gass SI & Fu MC (Eds), *Encyclopedia of operations research and management science*, Springer, New York (NY).

[61] Holland JH, 1992, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, Cambridge, Massachusetts.

[62] Homberger J & Gehring H, 1999, *Two evolutionary metaheuristics for the vehicle routing problem with time windows*, INFORMS Journal on Operations Research, **37(3)**, pp. 297–318.

[63] Hurricane Media, 2018, *Capacitated Vehicle Routing Problem Library*, [Online], [Cited August 2018], Available from `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`.

[64] Inflectra, 2018, *Software testing methodologies – Learn the methods & tools*, [Online], [Cited March 2018], Available from `https://www.inflectra.com/ideas/topic/testing-methodologies.aspx`.

[65] INKAS, 2018, *Armored cash-in-transit vehicle based on Hino 338*, [Online], [Cited October 2018], Available from `https://inkasarmored.com/armored-hino-338-cit/`.

[66] Jaillet P & Wagner MR, 2008, *Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses*, INFORMS Journal on Operations Research, **56(3)**, pp. 745–757.

[67] Keen PGW, 1980, *Adaptive design for decision support systems*, Association for Computing Machinery SIGOA Newsletter, **1(4-5)**, pp. 15–25.

[68] Keen PGW, 1987, *Decision support systems: The next decade*, Decision Support Systems, **3**, pp. 253–265.

[69] Kendall KE & Kendall JE, 2014, *Systems Analysis and Design*, 9th Edition, Pearson, New Jersey (NJ).

[70] Kilby P, Prosser P & Shaw P, 1999, *Guided local search for the vehicle routing problem with time windows*, Proceedings of the second International Conference on Metaheuristics, Glasgow, pp. 473–486.

[71] Kirkpatrick S, Gelatt CD & Vecchi MP, 1983, *Optimization by simulated annealing*, American Association for the Advancement of Science, **220**, pp. 671–680.

[72] Kontoravdis G & Bard J, 1995, *A GRASP for the vehicle routing problem with time windows*, ORSA Journal on Computing, **7(1)**, pp. 10–23.

[73] Kramer O, 2017, *Genetic algorithm essentials*, Springer, New York (NY).

[74] Kumar SN & Panneerselvam R, 2012, *A survey on the vehicle routing problem and its variants*, Intelligent Information Management, **4(3)**, pp. 66.

[75] Landry M, Malouin JL & Oral M, 1983, *Model validation in operations research*, European Journal of Operational Research, **14(3)**, pp. 207–220.

[76] Laporte G & Nobert Y, 1985, *Vehicle routing*, INFORMS Journal on Operations Research, **33(5)**, pp. 1050–1073.

[77] Larsen A, 2000, *The dynamic vehicle routing problem*, Phd dissertation, Technical University of Denmark, Denmark.

[78] Law A & Kelton W, 1991, *Simulation modelling and analysis*, 3rd Edition, McGaw-Hill, New York (NY).

[79] Leahy P, 2018, *What is Java?*, [Online], [Cited August 2018], Available from `https://www.thoughtco.com/what-is-java-2034117`.

[80] Lee T, 2018, *Optimisation versus Simulation*, [Online], [Cited November 2018], Available from `https://www.kisters.net/NA/fileadmin/KNA/Products/Optimization_vs_Simulation.pdf`.

[81] Lenstra JK & Kan AHG, 1981, *Complexity of vehicle routing and scheduling problems*, Networks, **11(2)**, pp. 221–227.

[82] Li CL, Simchi-Levi D & Desrochers M, 1992, *On the distance constrained vehicle routing problem*, Operations Research, **40(4)**, pp. 790–799.

[83] Lund K, Oli BGM & Rygaard JM, 1996, *Vehicle routing problems with varying degrees of dynamism*, Technical report, The Department of Mathematical Modelling, Technical University of Denmark.

[84] Malhotra MK, Sharma S & Nair SS, 1999, *Decision making using multiple models*, European Journal of Operational Research, **114(1)**, pp. 1–14.

[85] Management Study HQ, 2018, *Components of Decision Support Systems*, [Online], [Cited March 2018], Available from `https://www.managementstudyhq.com/components-of-decision-support-systems.html`.

[86] Maria A, 1997, *Introduction to modelling and simulation*, Proceedings of the twenty-ninth Winter Simulation Conference, New York, pp. 7–13.

[87] Martis MS, 2006, *Validation of simulation based models: A theoretical outlook*, The electronic journal of business research methods, **4(1)**, pp. 39–46.

[88] Masum AKM, Shahjalal M, Faruque F & Sarker IH, 2011, *Solving the vehicle routing problem using genetic algorithm*, International Journal of Advanced Computer Science and Applications, **2(7)**, pp. 126–131.

[89] Masweneng K, 2018, *Times Live*, [Online], [Cited May 2018], Available from `https://www.timeslive.co.za/news/south-africa/2018-05-18`.

[90] Maynard J, 2017, *How much money is floating around in notes and coins in SA?*, [Online], [Cited May 2018], Available from `https://za.investing.com/analysis/how-much-money-is-floating-around-in-notes-and-coins-in-sa-200189651`.

[91] Michallet J, Prins C, Amodeo L, Yalaoui F & Vitry G, 2014, *Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services*, Computers & Operations Research, **41**, pp. 196–207.

[92] Microsoft, 2018, *Microsoft Excel*, [Online], [Cited September 2018], Available from `https://products.office.com/en-za/excel`.

[93] Microsoft, 2018, *Microsoft Word*, [Online], [Cited August 2018], Available from `https://products.office.com/en-za/word`.

[94] Miller CE, Tucker AW & Zemlin RA, 1960, *Integer programming formulation of traveling salesman problems*, Journal of the Association of Computing Machinery, **7(4)**, pp. 326–329.

[95] Naidoo P, 2017, *Almost 1 cash-in-transit incident experienced per day in 2017*, [Online], [Cited December 2018], Available from `https://www.moneyweb.co.za/news/industry/almost-1-cash-in-transit-incident-experienced-per-day-in-2017/`.

[96] Networking and Emerging Optimization, 2013, *Solution methods for VRP*, [Online], [Cited May 2017], Available from `http://neo.lcc.uma.es/vrp/solution-methods/`.

[97] News24, 2018, *News24*, [Online], [Cited October 2018], Available from `https://www.news24.com/`.

[98] Ngueveu SU, Prins C & Wolfler Calvo R, 2010, *Lower and upper bounds for the m-peripatetic vehicle routing problem*, 4OR, **8(4)**, pp. 387–406.

[99] Nguyen AT, Reiter S & Rigo P, 2014, *A review on simulation-based optimization methods applied to building performance analysis*, Applied Energy, **113**, pp. 1043–1058.

[100] Obbayi SR, 2018, *Types of Database Management Systems*, [Online], [Cited June 2017], Available from `https://www.brighthub.com/internet/web-development/articles/110654.aspx`.

[101] Operations Research Group Bologna, 2018, *VRPLIB: A Vehicle Routing Problem Library*, [Online], [Cited August 2018], Available from `http://or.dei.unibo.it/library/vrplib-vehicle-routing-problem-library`.

[102] Osman IH, 1993, *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem*, Annals of Operations Research, **41(4)**, pp. 421–451.

[103] Paessens H, 1988, *The savings algorithm for the vehicle routing problem*, European Journal of Operational Research, **34(3)**, pp. 336–344.

[104] Panwar A, 2018, *Types of Database Management Systems*, [Online], [Cited November 2011], Available from `https://www.c-sharpcorner.com/UploadFile/65fc13/types-of-database-management-systems/`.

[105] Pasterfield K, 2017, *The rise of immersive learning*, [Online], [Cited November 2018], Available from `https://trainingindustry.com/magazine/nov-dec-2017/the-rise-of-immersive-learning/`.

[106] Pataki G, 2003, *Teaching integer programming formulations using the traveling salesman problem*, SIAM review, **45(1)**, pp. 116–123.

[107] Pichpibul T & Kawtummachai R, 2012, *An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem*, ScienceAsia, **38(3)**, pp. 307–318.

[108] Pillac V, Gendreau M, Gueret C & Medaglia AL, 2013, *A review of dynamic vehicle routing problems*, European Journal of Operational Research, **225(1)**, pp. 1–11.

[109] Pirim H, Bayraktar E & Eksioglu B, 2008, *Tabu Search: A comparative study*, pp. 1–28 in Jaziri W (Ed), *Tabu Search*, InTech, Croatia.

[110] Potvin JY & Bengio S, 1996, *The vehicle routing problem with time windows part II: Genetic search*, INFORMS Journal on Computing, **8(2)**, pp. 165–172.

[111] Potvin JY, Kervahut T, Garcia BL & Rousseau JM, 1996, *The vehicle routing problem with time windows part I: Tabu search*, INFORMS Journal on Computing, **8(2)**, pp. 158–164.

[112] Psaraftis HN, 1995, *Dynamic vehicle routing: Status and prospects*, Annals of Operations Research, **61(1)**, pp. 143–164.

[113] Putano B, 2017, *Most Popular and Influential Programming Languages of 2018*, [Online], [Cited December 2018], Available from `https://stackify.com/popular-programming-languages-2018/`.

[114] Rouse M, 2018, *Discrete event simulation (DES)*, [Online], [Cited October 2018], Available from `https://whatis.techtarget.com/definition/discrete-event-simulation-DES`.

[115] Russo F & Rindone C, 2011, *Planning in road evacuation: Classification of exogenous activities*, WIT Transactions on the Built Environment, **116**, pp. 639–651.

[116] Sargent RG, 2009, *Verification and validation of simulation models*, Proceedings of the Winter Simulation Conference (WSC), Austin, pp. 162–176.

[117] Sesant S, 2018, *Pictures: Number of cash-in-transit heists triple*, [Online], [Cited May 2017], Available from `https://www.iol.co.za/capeargus/news/pictures-number-of-cash-in-transit-heists-triple`.

[118] Shaw P, 1998, *Using constraint programming and local search methods to solve vehicle routing problems*, Proceedings of the fourth International conference on principles and practice of constraint programming, Glasgow, pp. 417–431.

[119] SlideShare, 2008, *Introduction to Simulation*, [Online], [Cited July 2017], Available from `https://www.slideshare.net/chimco.net/introduction-to-simulation-presentation`.

[120] Smith L & Louis E, 2017, *Cash-in-transit armed robbery in Australia*, [Online], [Cited November 2018], Available from `https://aic.gov.au/publications/tandi/tandi397`.

[121] Smith L & Louis E, 2010, *Cash in transit armed robbery in Australia*, Trends and Issues in Crime and Criminal Justice, **397**, pp. 1.

[122] Solomon MM, 1987, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations Research, **35(2)**, pp. 254–265.

[123] Soltesz DL, 2018, *Difference between flat file & Relational Database*, [Online], [Cited March 2018], Available from `https://www.techwalla.com/articles/difference-flat-file-relational-database`.

[124] Spivey MZ & Powell WB, 2004, *The dynamic assignment problem*, Transportation Science, **38(4)**, pp. 399–419.

[125] Taillard E, Badeau P, Gendreau M, Guertin F & Potvin JY, 1997, *A tabu search heuristic for the vehicle routing problem with soft time windows*, Transportation science, **31(2)**, pp. 170–186.

[126] Talarico L, Sorensen K & Springael J, 2015, *Metaheuristics for the risk-constrained cash-in-transit vehicle routing problem*, European Journal of Operational Research, **244(2)**, pp. 457–470.

[127] Taute BJ, 1983, *Quality assurance and maintenance application systems*, Proceedings of the AFIPS National Computer Conference, New York, pp. 123–129.

[128] Taylor JH, 2001, *Modelling & Simulation of Dynamic Systems — A tutorial*, Proceedings of the fourth International Symposium on Mathematical Modelling and Simulation in Agricultural and Bio-Industries, Plenary lecture, Israel, pp. 1–12.

[129] TechoPedia, 2018, *Extensible Markup Language (XML)*, [Online], [Cited June 2018], Available from `https://www.techopedia.com/definition/24387/extensible-markup-language-xml`.

[130] Techopedia, 2018, *Entity-Relationship Diagram (ERD)*, [Online], [Cited June 2018], Available from `https://www.techopedia.com/definition/1200/entity-relationship-diagram-erd`.

[131] TechTarget, 2018, *Comma-separated values file (CSV)*, [Online], [Cited June 2018], Available from `https://searchsqlserver.techtarget.com/definition/comma-separated-values-file`.

[132] TechTerms, 2018, *SQL Definition*, [Online], [Cited June 2018], Available from `https://techterms.com/definition/sql`.

[133] Teodorovic D & Pavkovic G, 1992, *A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand*, Transportation Planning and Technology, **16(4)**, pp. 261–273.

[134] Thangiah SR, Nygard K & Juell P, 1991, *GIDEON: A genetic algorithm system for vehicle routing with time windows*, Proceedings of the seventh IEEE Conference on Artificial Intelligence Applications, Berlin, pp. 322–328.

[135] Thangiah SR, Potvin JY & Sun T, 1996, *Heuristic approaches to vehicle routing with back hauls and time windows*, Computers & Operations Research, **23(11)**, pp. 1043–1057.

[136] Toth P & Vigo D, 2002, *Models, relaxations and exact approaches for the capacitated vehicle routing problem*, Discrete Applied Mathematics, **123(1)**, pp. 487–512.

[137] Training Industry, 2018, *Simulation Training*, [Online], [Cited October 2018], Available from `https://trainingindustry.com/glossary/simulation-training/`.

[138] Tsang E, 2005, *Combinatorial explosion*, [Online], [Cited May 2018], Available from `https://cswww.essex.ac.uk/CSP/ComputationalFinanceTeaching/CombinatorialExplosion.html`.

[139] Van Breedam A, 1996, *An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem*, RUCA working paper, University of Antwerp, Belgium.

[140] Van Raemdonck K, Macharis C & Mairesse O, 2013, *Risk analysis system for the transport of hazardous materials*, Journal of Safety Research, **45**, pp. 55–63.

[141] Vaughn PJ, 2001, *System implementation success factors — it is not just the technology*, Proceedings of the fifteenth CUMREC Conference, Québec, Canada, pp. 1–19.

[142] Vigeh A, 2011, *Investigation of a Simulated Annealing Cooling Schedule to Optimise the Estimation of the Fibre Diameter Distribution in a Peripheral Nerve Trunk*, MSc Thesis, California Polytechnic State University, San Luis Obispo (CA).

[143] Walters S, Broady J & Hartley R, 1994, *A review of information systems development methodologies*, Library management, **15(6)**, pp. 5–19.

[144] Wang L & Tan KC, 2006, *Graphical User Interface Design*, pp. 53–57 in Wiley (Ed), *Modern Industrial Automation Software Design*, Wiley Online Library, New Jersey (NJ).

[145] Watt A & Eng N, 2014, *Database design*, 2nd Edition, Open Educational Resources, Montreal.

[146] Willard J, 1989, *Vehicle routing using r-optimal tabu search*, Master's thesis, The Management School, Imperial College, London.

[147] Winston WL, 2004, *Operations research: Applications and algorithms*, Brooks/Cole, Belmont (CA).

[148] Wolfram, 2018, *Relative error*, [Online], [Cited November 2018], Available from `http://mathworld.wolfram.com/RelativeError.html`.

[149] Yan S, Chi CJ & Tang CH, 2006, *Inter-city bus routing and timetable setting under stochastic demands*, Transportation Research Part A: Policy and Practice, **40(7)**, pp. 572–586.

[150] Yan S, Wang SS & Wu MW, 2012, *A model with a solution algorithm for the cash transportation vehicle routing and scheduling problem*, Computers & Industrial Engineering, **63(2)**, pp. 464–473.

[151] Yellow P, 1970, *A computational modification to the savings method of vehicle scheduling*, Journal of the Operational Research Society, **21**, pp. 281–283.

[152] Yun HY, Jeong SJ & Kim KS, 2013, *Advanced harmony search with ant colony optimization for solving the traveling salesman problem*, Journal of Applied Mathematics, **2013**, pp. 1–8.

[153] Zhang CY, Li PG, Guan ZL & Rao YQ, 2007, *A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem*, Computers & Operations Research, **34(1)**, pp. 3229–3242.

APPENDIX A

Appendix A

Tables A.1–A.7 contain the results of the parameter variation performed on the RCTVRP model for the thirteen problem instances from the benchmark VRP library [63] with the twenty nine different risk threshold levels from Table 6.2. The left-most column indicates the iteration number. Let k denote the number of vehicles used to solve the problem, let d denote the total distance travelled by the fleet of vehicles, let Avg R denote the average risk anticipated for all routes in the solution and let RF (avg) denote the risk factor increase for the average route in the solution. Furthermore, let Max R denote risk most risk that is anticipated for a single route in the solution, let RF (max) denote the risk factor increase for the route with the highest risk in the solution and, finally let $\%R$ (max) denote the similarity between the route with the maximum anticipated risk along a route and the minimum allowable risk threshold (as explained in §6.2.2).

Problem instance	E_n22_k4								E_n23_k3							
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)		k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	
1	10	682	46543	0.60	77102	1.00	1.00		5	770	110959	0.81	137457	1.00	1.00	
2	8	607	77074	1.00	114831	1.49	1.01		4	686	150593	1.10	204245	1.49	1.01	
3	6	498	127279	1.65	145216	1.88	1.06		4	669	166576	1.21	263612	1.92	1.04	
4	5	463	169850	2.20	179706	2.33	1.07		3	624	268402	1.95	335154	2.44	1.02	
5	5	420	170314	2.21	230159	2.99	1.00		3	624	268402	1.95	335154	2.44	1.19	
6	5	402	176482	2.29	262621	3.41	1.03		3	626	268147	1.95	470906	3.43	1.02	
7	4	389	240085	3.11	276412	3.59	1.10		3	621	285536	2.08	538708	3.92	1.02	
8	4	389	240085	3.11	276412	3.59	1.20		3	621	285536	2.08	538708	3.92	1.13	
9	4	389	240085	3.11	276412	3.59	1.28		3	621	285536	2.08	538708	3.92	1.22	
10	4	389	240085	3.11	276412	3.59	1.35		3	621	285536	2.08	538708	3.92	1.29	
11	4	389	240085	3.11	276412	3.59	1.40		3	621	285536	2.08	538708	3.92	1.35	
12	4	389	240085	3.11	276412	3.59	1.45		3	621	285536	2.08	538708	3.92	1.40	
13	4	389	240085	3.11	276412	3.59	1.49		3	621	285536	2.08	538708	3.92	1.44	
14	4	389	240085	3.11	276412	3.59	1.52		3	621	285536	2.08	538708	3.92	1.48	
15	4	389	240085	3.11	276412	3.59	1.55		3	621	285536	2.08	538708	3.92	1.51	
16	4	389	240085	3.11	276412	3.59	1.58		3	621	285536	2.08	538708	3.92	1.54	
17	4	389	240085	3.11	276412	3.59	1.60		3	621	285536	2.08	538708	3.92	1.56	
18	4	389	240085	3.11	276412	3.59	1.62		3	621	285536	2.08	538708	3.92	1.59	
19	4	389	240085	3.11	276412	3.59	1.64		3	621	285536	2.08	538708	3.92	1.61	
20	4	389	240085	3.11	276412	3.59	1.66		3	621	285536	2.08	538708	3.92	1.63	
21	4	389	240085	3.11	276412	3.59	1.67		3	621	285536	2.08	538708	3.92	1.64	
22	4	389	240085	3.11	276412	3.59	1.69		3	621	285536	2.08	538708	3.92	1.66	
23	4	389	240085	3.11	276412	3.59	1.70		3	621	285536	2.08	538708	3.92	1.67	
24	4	389	240085	3.11	276412	3.59	1.71		3	621	285536	2.08	538708	3.92	1.69	
25	4	389	240085	3.11	276412	3.59	1.72		3	621	285536	2.08	538708	3.92	1.70	
26	4	389	240085	3.11	276412	3.59	1.73		3	621	285536	2.08	538708	3.92	1.71	
27	4	389	240085	3.11	276412	3.59	1.74		3	621	285536	2.08	538708	3.92	1.72	
28	4	389	240085	3.11	276412	3.59	1.75		3	621	285536	2.08	538708	3.92	1.73	
29	4	389	240085	3.11	276412	3.59	1.76		3	621	285536	2.08	538708	3.92	1.74	

TABLE A.1: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n22_k4 and E_n23_k3 from the benchmark VRP library [63].*

Problem instance	E_n30_k3							E_n33_k4						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	5	679	147472	0.85	174042	1.00	1.00	8	1469	312233	0.87	357816	1.00	1.00
2	4	577	194473	1.12	257614	1.48	1.01	6	1135	438468	1.23	532238	1.49	1.01
3	4	520	198533	1.14	309363	1.78	1.11	5	1047	562735	1.57	715330	2.00	1.00
4	4	520	198533	1.14	309363	1.78	1.29	4	864	719119	2.01	872398	2.44	1.02
5	4	520	198533	1.14	309363	1.78	1.41	4	845	722813	2.02	1031621	2.88	1.04
6	4	520	198533	1.14	309363	1.78	1.49	4	845	722813	2.02	1031621	2.88	1.18
7	4	520	198533	1.14	309363	1.78	1.56	4	845	722813	2.02	1031621	2.88	1.28
8	4	520	198533	1.14	309363	1.78	1.60	4	845	722813	2.02	1031621	2.88	1.36
9	4	520	198533	1.14	309363	1.78	1.64	4	845	722813	2.02	1031621	2.88	1.42
10	4	520	198533	1.14	309363	1.78	1.68	4	845	722813	2.02	1031621	2.88	1.48
11	4	520	198533	1.14	309363	1.78	1.70	4	845	722813	2.02	1031621	2.88	1.52
12	4	520	198533	1.14	309363	1.78	1.73	4	845	722813	2.02	1031621	2.88	1.56
13	4	520	198533	1.14	309363	1.78	1.75	4	845	722813	2.02	1031621	2.88	1.59
14	4	520	198533	1.14	309363	1.78	1.76	4	845	722813	2.02	1031621	2.88	1.62
15	4	520	198533	1.14	309363	1.78	1.78	4	845	722813	2.02	1031621	2.88	1.64
16	4	520	198533	1.14	309363	1.78	1.79	4	845	722813	2.02	1031621	2.88	1.66
17	4	520	198533	1.14	309363	1.78	1.80	4	845	722813	2.02	1031621	2.88	1.68
18	4	520	198533	1.14	309363	1.78	1.81	4	845	722813	2.02	1031621	2.88	1.70
19	4	520	198533	1.14	309363	1.78	1.82	4	845	722813	2.02	1031621	2.88	1.71
20	4	520	198533	1.14	309363	1.78	1.83	4	845	722813	2.02	1031621	2.88	1.73
21	4	520	198533	1.14	309363	1.78	1.84	4	845	722813	2.02	1031621	2.88	1.74
22	4	520	198533	1.14	309363	1.78	1.85	4	845	722813	2.02	1031621	2.88	1.75
23	4	520	198533	1.14	309363	1.78	1.85	4	845	722813	2.02	1031621	2.88	1.76
24	4	520	198533	1.14	309363	1.78	1.86	4	845	722813	2.02	1031621	2.88	1.77
25	4	520	198533	1.14	309363	1.78	1.86	4	845	722813	2.02	1031621	2.88	1.78
26	4	520	198533	1.14	309363	1.78	1.87	4	845	722813	2.02	1031621	2.88	1.79
27	4	520	198533	1.14	309363	1.78	1.87	4	845	722813	2.02	1031621	2.88	1.79
28	4	520	198533	1.14	309363	1.78	1.88	4	845	722813	2.02	1031621	2.88	1.80
29	4	520	198533	1.14	309363	1.78	1.88	4	845	722813	2.02	1031621	2.88	1.81

TABLE A.2: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n30_k3 and E_n33_k4 from the benchmark VRP library [63].*

Problem instance	E_n51_k5								E_n76_k7							
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)		k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	
1	22	1338	676	0.76	884	1.00	1.00		28	1772	1035	0.77	884	0.66	1.00	
2	17	1121	1063	1.20	1350	1.53	0.98		21	1444	1781	1.33	1350	1.01	1.00	
3	14	976	1526	1.73	1802	2.04	0.98		17	1241	2407	1.80	1802	1.34	1.01	
4	12	886	1967	2.23	2271	2.57	0.97		14	1114	2816	2.10	2271	1.70	1.01	
5	10	794	2448	2.77	2667	3.02	0.99		13	1023	3511	2.62	2667	1.99	1.01	
6	9	736	2856	3.23	3144	3.56	0.98		11	980	3941	2.94	3144	2.35	1.00	
7	9	705	2967	3.36	3606	4.08	0.98		10	935	4618	3.45	3606	2.69	1.00	
8	8	690	3485	3.94	4013	4.54	0.99		10	874	5230	3.90	4013	3.00	1.00	
9	8	686	3597	4.07	4526	5.12	0.98		9	849	6363	4.75	4526	3.38	1.00	
10	7	690	4761	5.39	4994	5.65	0.97		9	842	6328	4.72	4994	3.73	1.01	
11	7	643	4449	5.03	5438	6.15	0.97		9	827	6174	4.61	5438	4.06	1.01	
12	7	633	4854	5.49	5894	6.67	0.97		8	823	6383	4.76	5894	4.40	1.01	
13	7	638	4754	5.38	6252	7.07	0.99		8	800	7638	5.70	6252	4.67	1.00	
14	6	604	5760	6.52	6683	7.56	0.99		7	798	7696	5.75	6683	4.99	1.02	
15	6	596	5777	6.54	7264	8.22	0.97		7	788	7893	5.89	7264	5.42	1.00	
16	6	600	5908	6.68	7696	8.71	0.98		7	782	7920	5.91	7696	5.74	1.00	
17	6	594	6011	6.80	8140	9.21	0.98		7	761	9653	7.21	8140	6.08	1.00	
18	6	593	6040	6.83	8603	9.73	0.98		7	744	10033	7.49	8603	6.42	1.01	
19	6	587	5986	6.77	8822	9.98	1.00		7	730	9628	7.19	8822	6.59	1.01	
20	5	602	7969	9.01	9400	10.63	0.99		7	730	9932	7.41	9400	7.02	1.00	
21	6	601	6286	7.11	9914	11.22	0.98		7	739	10191	7.61	9914	7.40	1.01	
22	6	600	6219	7.04	10034	11.35	1.01		7	739	10191	7.61	10034	7.49	1.05	
23	6	600	6254	7.07	10477	11.85	1.01		7	740	10366	7.74	10477	7.82	1.02	
24	6	600	6254	7.07	10477	11.85	1.05		7	740	10366	7.74	10477	7.82	1.06	
25	6	600	6254	7.07	10477	11.85	1.09		7	740	10366	7.74	10477	7.82	1.09	
26	6	600	6254	7.07	10477	11.85	1.12		7	740	10366	7.74	10477	7.82	1.13	
27	6	600	6254	7.07	10477	11.85	1.15		7	740	10366	7.74	10477	7.82	1.16	
28	6	600	6254	7.07	10477	11.85	1.18		7	740	10366	7.74	10477	7.82	1.19	
29	6	600	6254	7.07	10477	11.85	1.21		7	740	10366	7.74	10477	7.82	1.21	

TABLE A.3: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n51_k5 and E_n76_k7 from the benchmark VRP library [63].*

Problem instance	E_n76_k8							E_n76_k10						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	28	1756	1073	0.80	1340	1.00	1.00	28	1756	1073	0.80	1340	1.00	1.00
2	21	1435	1766	1.32	2010	1.50	1.00	21	1435	1766	1.32	2010	1.50	1.00
3	17	1228	2428	1.81	2671	1.99	1.00	17	1228	2428	1.81	2671	1.99	1.00
4	14	1115	3046	2.27	3330	2.49	1.01	14	1115	3046	2.27	3330	2.49	1.01
5	13	991	3450	2.58	3987	2.98	1.01	13	996	3454	2.58	3987	2.98	1.01
6	11	908	4272	3.19	4692	3.50	1.00	12	925	3900	2.91	4690	3.50	1.00
7	10	889	4968	3.71	5336	3.98	1.00	11	916	4545	3.39	5330	3.98	1.01
8	10	859	5046	3.77	6029	4.50	1.00	11	905	4604	3.44	5915	4.42	1.02
9	10	881	5439	4.06	6679	4.99	1.00	11	915	4935	3.68	6654	4.97	1.01
10	9	847	6498	4.85	7321	5.47	1.01	11	917	5143	3.84	7327	5.47	1.01
11	9	856	6686	4.99	8028	5.99	1.00	11	927	5172	3.86	7682	5.73	1.04
12	9	846	6552	4.89	8667	6.47	1.00	11	927	5172	3.86	7682	5.73	1.12
13	8	826	7849	5.86	9358	6.99	1.00	10	903	5760	4.30	9227	6.89	1.02
14	9	835	6810	5.08	9856	7.36	1.02	10	900	5787	4.32	9585	7.16	1.05
15	8	816	8168	6.10	10622	7.93	1.01	10	900	5787	4.32	9585	7.16	1.11
16	8	805	8014	5.98	11051	8.25	1.03	10	900	5787	4.32	9585	7.16	1.16
17	8	801	8272	6.18	11903	8.89	1.01	10	900	5787	4.32	9585	7.16	1.20
18	8	801	8272	6.18	11903	8.89	1.06	10	900	5787	4.32	9585	7.16	1.25
19	8	799	8330	6.22	12928	9.65	1.03	10	900	5787	4.32	9585	7.16	1.28
20	8	799	8330	6.22	12928	9.65	1.08	10	900	5787	4.32	9585	7.16	1.32
21	8	799	8330	6.22	12928	9.65	1.12	10	900	5787	4.32	9585	7.16	1.35
22	8	799	8330	6.22	12928	9.65	1.16	10	900	5787	4.32	9585	7.16	1.38
23	8	799	8330	6.22	12928	9.65	1.20	10	900	5787	4.32	9585	7.16	1.40
24	8	799	8330	6.22	12928	9.65	1.23	10	900	5787	4.32	9585	7.16	1.43
25	8	799	8330	6.22	12928	9.65	1.26	10	900	5787	4.32	9585	7.16	1.45
26	8	799	8330	6.22	12928	9.65	1.29	10	900	5787	4.32	9585	7.16	1.47
27	8	799	8330	6.22	12928	9.65	1.31	10	900	5787	4.32	9585	7.16	1.49
28	8	799	8330	6.22	12928	9.65	1.33	10	900	5787	4.32	9585	7.16	1.51
29	8	799	8330	6.22	12928	9.65	1.36	10	900	5787	4.32	9585	7.16	1.52

TABLE A.4: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n76_k8 and E_n76_k10 from the benchmark VRP library [63].*

Problem instance	E_n101_k8								E_n101_k14							
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)		k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	
1	31	2080	1127	0.86	1304	1.00	1.00		31	2080	1127	0.86	1304	1.00	1.00	
2	24	1703	1703	1.31	1973	1.51	0.99		24	1703	1703	1.31	1973	1.51	0.99	
3	19	1471	2432	1.86	2629	2.02	0.99		19	1471	2432	1.86	2629	2.02	0.99	
4	16	1283	3019	2.32	3290	2.52	0.99		17	1313	2806	2.15	3290	2.52	0.99	
5	14	1207	3607	2.77	3906	2.99	1.00		15	1242	3342	2.56	3937	3.02	0.99	
6	13	1141	4169	3.20	4609	3.53	0.99		14	1221	3929	3.01	4590	3.52	0.99	
7	12	1108	4814	3.69	5265	4.04	0.99		14	1182	3992	3.06	5252	4.03	0.99	
8	11	1023	5250	4.03	5919	4.54	0.99		14	1167	3964	3.04	5872	4.50	1.00	
9	11	1028	5560	4.26	6577	5.04	0.99		14	1148	4034	3.09	6547	5.02	1.00	
10	10	989	6362	4.88	7247	5.56	0.99		14	1139	4055	3.11	7134	5.47	1.01	
11	9	960	7378	5.66	7889	6.05	0.99		14	1139	4055	3.11	7134	5.47	1.09	
12	9	955	7550	5.79	8551	6.56	0.99		14	1139	4055	3.11	7134	5.47	1.16	
13	8	937	8750	6.71	9183	7.04	0.99		14	1139	4055	3.11	7134	5.47	1.22	
14	9	954	7975	6.12	9814	7.53	1.00		14	1139	4055	3.11	7134	5.47	1.27	
15	8	910	8734	6.70	10462	8.02	1.00		14	1139	4055	3.11	7134	5.47	1.32	
16	8	903	9041	6.93	11097	8.51	1.00		14	1139	4055	3.11	7134	5.47	1.36	
17	8	913	9478	7.27	11813	9.06	0.99		14	1139	4055	3.11	7134	5.47	1.39	
18	8	912	9443	7.24	12410	9.52	1.00		14	1139	4055	3.11	7134	5.47	1.42	
19	8	901	9628	7.38	13090	10.04	1.00		14	1139	4055	3.11	7134	5.47	1.45	
20	8	895	9389	7.20	13461	10.32	1.02		14	1139	4055	3.11	7134	5.47	1.48	
21	8	892	9679	7.42	14033	10.76	1.02		14	1139	4055	3.11	7134	5.47	1.50	
22	8	892	9679	7.42	14033	10.76	1.06		14	1139	4055	3.11	7134	5.47	1.52	
23	8	890	9865	7.56	15217	11.67	1.03		14	1139	4055	3.11	7134	5.47	1.54	
24	8	890	9865	7.56	15217	11.67	1.07		14	1139	4055	3.11	7134	5.47	1.56	
25	8	890	9865	7.56	15217	11.67	1.10		14	1139	4055	3.11	7134	5.47	1.58	
26	8	890	9865	7.56	15217	11.67	1.14		14	1139	4055	3.11	7134	5.47	1.59	
27	8	890	9865	7.56	15217	11.67	1.17		14	1139	4055	3.11	7134	5.47	1.61	
28	8	890	9865	7.56	15217	11.67	1.20		14	1139	4055	3.11	7134	5.47	1.62	
29	8	890	9865	7.56	15217	11.67	1.22		14	1139	4055	3.11	7134	5.47	1.64	

TABLE A.5: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n101_k8 and E_n101_k14 from the benchmark VRP library [63].*

Problem instance	E_n121_k7							E_n151_k12						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	37	5366	1529	0.77	1987	1.00	1.00	47	2988	1163	0.88	1317	1.00	1.00
2	27	4149	2621	1.32	2983	1.50	1.00	34	2282	1823	1.38	1975	1.50	1.00
3	22	3160	3533	1.78	3978	2.00	1.00	27	1898	2442	1.85	2632	2.00	1.00
4	18	2663	4460	2.25	4973	2.50	1.00	23	1730	3077	2.34	3295	2.50	1.00
5	15	2326	5454	2.75	5966	3.00	1.00	20	1572	3786	2.87	3942	2.99	1.00
6	13	2078	6548	3.30	6962	3.50	1.00	19	1486	3997	3.04	4608	3.50	1.00
7	12	1908	7240	3.64	7945	4.00	1.00	16	1359	5085	3.86	5272	4.00	1.00
8	11	1739	7999	4.03	8949	4.50	1.00	15	1302	5619	4.27	5921	4.50	1.00
9	10	1577	8691	4.37	9944	5.01	1.00	14	1294	6369	4.84	6564	4.98	1.00
10	10	1557	9086	4.57	10923	5.50	1.00	13	1236	7040	5.35	7243	5.50	1.00
11	10	1484	9381	4.72	11933	6.01	1.00	13	1146	6648	5.05	7889	5.99	1.00
12	9	1418	10550	5.31	12925	6.51	1.00	13	1156	6789	5.16	8566	6.50	1.00
13	9	1379	10955	5.51	13907	7.00	1.00	12	1143	7756	5.89	9190	6.98	1.00
14	8	1333	12753	6.42	14922	7.51	1.00	12	1148	8158	6.20	9851	7.48	1.00
15	9	1304	11046	5.56	15892	8.00	1.00	12	1117	7766	5.90	10516	7.99	1.00
16	8	1301	12808	6.45	16912	8.51	1.00	12	1109	7956	6.04	11025	8.37	1.02
17	8	1261	12960	6.52	17904	9.01	1.00	12	1156	8453	6.42	11737	8.91	1.01
18	8	1209	13139	6.61	18866	9.50	1.00	12	1155	8534	6.48	12401	9.42	1.01
19	7	1195	15676	7.89	19709	9.92	1.01	12	1155	8534	6.48	12401	9.42	1.06
20	7	1098	14954	7.53	20678	10.41	1.01	12	1155	8534	6.48	12401	9.42	1.10
21	7	1098	15068	7.58	21027	10.58	1.04	12	1155	8534	6.48	12401	9.42	1.14
22	7	1098	15068	7.58	21027	10.58	1.08	12	1155	8534	6.48	12401	9.42	1.18
23	7	1098	15068	7.58	21027	10.58	1.12	12	1155	8534	6.48	12401	9.42	1.22
24	7	1098	15068	7.58	21027	10.58	1.15	12	1155	8534	6.48	12401	9.42	1.25
25	7	1098	15068	7.58	21027	10.58	1.19	12	1155	8534	6.48	12401	9.42	1.28
26	7	1098	15068	7.58	21027	10.58	1.22	12	1155	8534	6.48	12401	9.42	1.30
27	7	1098	15068	7.58	21027	10.58	1.24	12	1155	8534	6.48	12401	9.42	1.33
28	7	1098	15068	7.58	21027	10.58	1.27	12	1155	8534	6.48	12401	9.42	1.35
29	7	1098	15068	7.58	21027	10.58	1.29	12	1155	8534	6.48	12401	9.42	1.37

TABLE A.6: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n121_k7 and E_n151_k12 from the benchmark VRP library [63].*

Problem instance	E_n200_k16						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	64	3917	1179	0.88	1334	1.00	1.00
2	45	2987	1883	1.41	2001	1.50	1.00
3	36	2424	2477	1.86	2667	2.00	1.00
4	30	2088	3075	2.30	3314	2.48	1.01
5	26	1922	3715	2.79	4002	3.00	1.00
6	24	1819	4219	3.16	4664	3.50	1.00
7	22	1745	4976	3.73	5332	4.00	1.00
8	20	1583	5377	4.03	5974	4.48	1.00
9	19	1549	5942	4.45	6670	5.00	1.00
10	18	1532	6526	4.89	7315	5.48	1.00
11	17	1457	6967	5.22	7958	5.97	1.01
12	17	1429	7167	5.37	8667	6.50	1.00
13	17	1438	7273	5.45	9233	6.92	1.01
14	17	1415	7304	5.48	9995	7.49	1.00
15	17	1439	7516	5.63	10603	7.95	1.01
16	17	1443	7567	5.67	11267	8.45	1.01
17	16	1402	7983	5.98	11901	8.92	1.01
18	16	1419	8235	6.17	12416	9.31	1.02
19	16	1415	8211	6.16	13302	9.97	1.00
20	17	1422	7695	5.77	13805	10.35	1.01
21	17	1422	7695	5.77	13805	10.35	1.06
22	17	1418	7704	5.78	15027	11.26	1.02
23	17	1419	7729	5.79	15699	11.77	1.02
24	17	1419	7729	5.79	15699	11.77	1.06
25	17	1419	7729	5.79	15699	11.77	1.09
26	17	1419	7729	5.79	15699	11.77	1.13
27	17	1419	7729	5.79	15699	11.77	1.16
28	17	1419	7729	5.79	15699	11.77	1.19
29	17	1419	7729	5.79	15699	11.77	1.22

TABLE A.7: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n200_k16 from the benchmark VRP library [63].*

APPENDIX B

Appendix B

Tables B.1–B.7 contain the results of the parameter variation performed on the RCTVRP model for the thirteen problem instances from the benchmark VRP library [63] with the twenty nine different risk threshold levels from Table 6.2, without capacity constraints. The left-most column indicates the iteration number. Let k denote the number of vehicles used to solve the problem, let d denote the total distance travelled by the fleet of vehicles, let Avg R denote the average risk anticipated for all routes in the solution and let RF (avg) denote the risk factor increase for the average route in the solution. Furthermore, let Max R denote the most risk that is anticipated for a single route in the solution, let RF (max) denote the risk factor increase for the route with the highest risk in the solution and, finally let $\%R$ (max) denote the similarity between the route with the maximum anticipated risk along a route and the minimum allowable risk threshold (as explained in §6.2.2).

Problem instance	E_n22_k4							E_n23_k3						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	10	682	46543	0.60	77102	1.00	1.00	5	770	110959	0.81	137457	1.00	1.00
2	8	607	77074	1.00	114831	1.49	1.01	4	681	150290	1.09	187786	1.37	1.09
3	6	498	127279	1.65	145216	1.88	1.06	3	631	231590	1.68	270105	1.97	1.02
4	5	463	169850	2.20	179706	2.33	1.07	3	627	216159	1.57	311957	2.27	1.09
5	5	416	181494	2.35	230159	2.99	1.00	2	556	389648	2.83	399422	2.91	1.03
6	4	383	220763	2.86	262621	3.41	1.03	2	556	389648	2.83	399422	2.91	1.17
7	4	384	225475	2.92	297854	3.86	1.03	2	556	389648	2.83	399422	2.91	1.27
8	4	383	227443	2.95	330604	4.29	1.05	2	556	389648	2.83	399422	2.91	1.35
9	4	401	286002	3.71	378831	4.91	1.02	2	556	389648	2.83	399422	2.91	1.42
10	3	369	396346	5.14	422729	5.48	1.00	2	556	389648	2.83	399422	2.91	1.47
11	3	362	397892	5.16	446188	5.79	1.04	2	556	389648	2.83	399422	2.91	1.52
12	3	340	377507	4.90	500948	6.50	1.00	2	556	389648	2.83	399422	2.91	1.55
13	3	329	388194	5.03	534615	6.93	1.01	2	556	389648	2.83	399422	2.91	1.58
14	3	326	396201	5.14	552192	7.16	1.05	2	556	389648	2.83	399422	2.91	1.61
15	3	326	396201	5.14	552192	7.16	1.10	2	556	389648	2.83	399422	2.91	1.64
16	3	329	420519	5.45	624062	8.09	1.05	2	556	389648	2.83	399422	2.91	1.66
17	3	322	422697	5.48	655386	8.50	1.06	2	556	389648	2.83	399422	2.91	1.68
18	3	322	422697	5.48	655386	8.50	1.11	2	556	389648	2.83	399422	2.91	1.69
19	3	322	422697	5.48	655386	8.50	1.15	1	550	1370524	9.97	1370524	9.97	1.00
20	3	322	422697	5.48	655386	8.50	1.19	1	550	1370524	9.97	1370524	9.97	1.05
21	3	322	422697	5.48	655386	8.50	1.23	1	536	1448268	10.54	1448268	10.54	1.04
22	3	322	422697	5.48	655386	8.50	1.26	1	536	1448268	10.54	1448268	10.54	1.08
23	3	322	422697	5.48	655386	8.50	1.29	1	536	1448268	10.54	1448268	10.54	1.12
24	2	302	795909	10.32	936433	12.15	1.03	2	601	899285	6.54	1696599	12.34	1.01
25	2	302	795909	10.32	936433	12.15	1.07	2	601	899285	6.54	1696599	12.34	1.05
26	2	302	795909	10.32	936433	12.15	1.10	2	590	953234	6.93	1824460	13.27	1.02
27	2	302	795909	10.32	936433	12.15	1.13	2	590	953234	6.93	1824460	13.27	1.05
28	2	302	795909	10.32	936433	12.15	1.16	2	580	1004262	7.31	1979185	14.40	1.01
29	2	302	795909	10.32	936433	12.15	1.19	2	565	1014400	7.38	1999461	14.55	1.03

TABLE B.1: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n22_k4 and E_n23_k3 from the benchmark VRP library [63], without the capacity constraint.*

Problem instance	E_n30_k3							E_n33_k4						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	5	679	147472	0.85	174042	1.00	1.00	8	1469	312233	0.87	357816	1.00	1.00
2	4	577	194473	1.12	257614	1.48	1.01	6	1135	438468	1.23	532238	1.49	1.01
3	3	520	318880	1.83	338035	1.94	1.03	5	1047	562735	1.57	715330	2.00	1.00
4	3	500	311298	1.79	379467	2.18	1.13	4	875	739630	2.07	893393	2.50	1.00
5	3	500	311298	1.79	379467	2.18	1.27	3	702	964319	2.70	1064234	2.97	1.01
6	3	513	334280	1.92	563675	3.24	1.07	3	737	1150127	3.21	1238894	3.46	1.01
7	2	444	627251	3.60	686673	3.95	1.01	3	728	1098570	3.07	1413448	3.95	1.01
8	2	414	553377	3.18	727286	4.18	1.07	3	684	1123393	3.14	1571575	4.39	1.02
9	2	413	586513	3.37	793558	4.56	1.09	3	683	1145908	3.20	1735048	4.85	1.03
10	2	413	586513	3.37	793558	4.56	1.17	3	653	1250056	3.49	1914545	5.35	1.03
11	2	413	586513	3.37	793558	4.56	1.24	2	592	1990068	5.56	2130408	5.95	1.01
12	2	413	586513	3.37	793558	4.56	1.30	2	608	2094158	5.85	2252009	6.29	1.03
13	2	413	586513	3.37	793558	4.56	1.35	2	605	2018030	5.64	2497047	6.98	1.00
14	2	413	586513	3.37	793558	4.56	1.39	2	607	2024066	5.66	2576054	7.20	1.04
15	2	413	586513	3.37	793558	4.56	1.43	2	604	2097736	5.86	2700544	7.55	1.06
16	2	413	586513	3.37	793558	4.56	1.46	2	605	2147155	6.00	2926834	8.18	1.04
17	2	413	586513	3.37	793558	4.56	1.49	2	594	2217475	6.20	3202568	8.95	1.01
18	2	413	586513	3.37	793558	4.56	1.52	2	594	2217475	6.20	3202568	8.95	1.06
19	2	413	586513	3.37	793558	4.56	1.54	2	594	2217475	6.20	3202568	8.95	1.10
20	2	413	586513	3.37	793558	4.56	1.57	2	594	2135197	5.97	3749274	10.48	1.00
21	2	413	586513	3.37	793558	4.56	1.59	2	583	2150242	6.01	3859167	10.79	1.02
22	2	413	586513	3.37	793558	4.56	1.60	2	583	2150242	6.01	3859167	10.79	1.06
23	2	413	586513	3.37	793558	4.56	1.62	2	594	2357529	6.59	4287662	11.98	1.00
24	2	413	586513	3.37	793558	4.56	1.64	2	609	2375131	6.64	4452819	12.44	1.00
25	2	413	586513	3.37	793558	4.56	1.65	2	595	2507931	7.01	4641928	12.97	1.00
26	2	413	586513	3.37	793558	4.56	1.66	2	573	2480577	6.93	4698806	13.13	1.03
27	1	409	2414695	13.87	2414695	13.87	1.01	2	573	2480577	6.93	4698806	13.13	1.06
28	1	406	2495221	14.34	2495221	14.34	1.01	2	583	2709123	7.57	5173330	14.46	1.00
29	1	406	2495221	14.34	2495221	14.34	1.04	2	583	2709123	7.57	5173330	14.46	1.04

TABLE B.2: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n30_k3 and E_n33_k4 from the benchmark VRP library [63], without the capacity constraint.*

Problem instance	E_n51_k5								E_n76_k7							
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)		k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	
1	22	1338	676	0.76	884	1.00	1.00		28	1772	1035	0.77	1341	1.00	1.00	
2	17	1121	1063	1.20	1350	1.53	0.98		21	1444	1781	1.33	2010	1.50	1.00	
3	14	976	1526	1.73	1802	2.04	0.98		17	1241	2407	1.80	2656	1.98	1.01	
4	12	886	1967	2.23	2271	2.57	0.97		15	1114	2816	2.10	3327	2.48	1.01	
5	10	794	2448	2.77	2667	3.02	0.99		13	1023	3511	2.62	3989	2.98	1.01	
6	9	736	2856	3.23	3144	3.56	0.98		12	980	3941	2.94	4684	3.50	1.00	
7	9	705	2967	3.36	3606	4.08	0.98		11	935	4618	3.45	5336	3.98	1.00	
8	8	690	3485	3.94	4013	4.54	0.99		10	874	5230	3.90	6016	4.49	1.00	
9	8	686	3597	4.07	4526	5.12	0.98		9	849	6363	4.75	6699	5.00	1.00	
10	7	690	4761	5.39	4994	5.65	0.97		9	842	6328	4.72	7321	5.46	1.01	
11	7	643	4449	5.03	5438	6.15	0.97		9	827	6174	4.61	7997	5.97	1.01	
12	7	633	4854	5.49	5894	6.67	0.97		9	823	6383	4.76	8662	6.47	1.01	
13	7	638	4754	5.38	6252	7.07	0.99		8	800	7638	5.70	9341	6.97	1.00	
14	6	604	5760	6.52	6683	7.56	0.99		7	792	9385	7.01	9835	7.34	1.02	
15	6	596	5777	6.54	7264	8.22	0.97		8	788	7825	5.84	10687	7.98	1.00	
16	6	600	5908	6.68	7696	8.71	0.98		7	772	9519	7.11	11366	8.48	1.00	
17	6	589	5861	6.63	8140	9.21	0.98		7	761	9657	7.21	12044	8.99	1.00	
18	6	588	5890	6.66	8603	9.73	0.98		7	735	10358	7.73	12673	9.46	1.00	
19	5	555	7077	8.01	9056	10.24	0.98		6	722	12247	9.14	13312	9.94	1.01	
20	5	583	7584	8.58	9400	10.63	0.99		6	722	12702	9.48	14012	10.46	1.00	
21	5	587	7956	9.00	9993	11.30	0.97		6	730	12599	9.40	14579	10.88	1.01	
22	5	566	7338	8.30	10277	11.63	0.99		6	728	12505	9.33	14791	11.04	1.04	
23	5	572	8085	9.15	10871	12.30	0.98		6	702	12878	9.61	16065	11.99	1.00	
24	5	571	8130	9.20	10936	12.37	1.01		6	696	12797	9.55	16161	12.06	1.03	
25	5	567	8734	9.88	11598	13.12	0.99		5	683	16190	12.09	17397	12.99	1.00	
26	4	558	11747	13.29	12250	13.86	0.97		6	693	13117	9.79	17826	13.31	1.01	
27	4	576	12231	13.84	12548	14.20	0.99		5	663	15816	11.81	18726	13.98	1.00	
28	4	564	11888	13.45	13009	14.72	0.99		5	660	16091	12.01	18952	14.15	1.02	
29	4	564	11826	13.38	13520	15.30	0.98		5	674	16843	12.57	20014	14.94	1.00	

TABLE B.3: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n51_k5 and E_n76_k7 from the benchmark VRP library [63], without the capacity constraint.*

Problem instance	E_n76_k8						E_n76_k10					
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	28	1756	1073	0.80	1340	1.00	1.00	28	1756	1073	0.80	1340	1.00	1.00
2	21	1435	1766	1.32	2010	1.50	1.00	21	1435	1766	1.32	2010	1.50	1.00
3	17	1228	2428	1.81	2671	1.99	1.00	17	1228	2428	1.81	2671	1.99	1.00
4	14	1115	3046	2.27	3330	2.49	1.01	14	1115	3046	2.27	3330	2.49	1.01
5	13	991	3450	2.58	3987	2.98	1.01	13	991	3450	2.58	3987	2.98	1.01
6	11	908	4272	3.19	4692	3.50	1.00	11	908	4272	3.19	4692	3.50	1.00
7	10	889	4968	3.71	5336	3.98	1.00	10	889	4968	3.71	5336	3.98	1.00
8	10	859	5046	3.77	6029	4.50	1.00	10	859	5046	3.77	6029	4.50	1.00
9	9	877	6330	4.73	6679	4.99	1.00	9	877	6330	4.73	6679	4.99	1.00
10	9	843	6470	4.83	7321	5.47	1.01	9	843	6470	4.83	7321	5.47	1.01
11	9	853	6849	5.11	8028	5.99	1.00	9	853	6849	5.11	8028	5.99	1.00
12	8	816	7637	5.70	8687	6.48	1.00	8	816	7637	5.70	8687	6.48	1.00
13	8	817	8095	6.04	9358	6.99	1.00	8	817	8095	6.04	9358	6.99	1.00
14	7	768	9425	7.04	10055	7.51	1.00	7	768	9425	7.04	10055	7.51	1.00
15	7	775	9399	7.02	10637	7.94	1.01	7	775	9399	7.02	10637	7.94	1.01
16	7	740	9087	6.78	11392	8.50	1.00	7	740	9087	6.78	11392	8.50	1.00
17	7	729	9228	6.89	12033	8.98	1.00	7	729	9228	6.89	12033	8.98	1.00
18	7	722	9105	6.80	12251	9.15	1.04	7	722	9105	6.80	12251	9.15	1.04
19	6	723	12111	9.04	13374	9.98	1.00	6	723	12111	9.04	13374	9.98	1.00
20	6	723	12020	8.97	14071	10.50	1.00	6	723	12020	8.97	14071	10.50	1.00
21	6	711	12072	9.01	14589	10.89	1.01	6	711	12072	9.01	14589	10.89	1.01
22	6	702	12084	9.02	15319	11.44	1.01	6	702	12084	9.02	15319	11.44	1.01
23	6	712	12228	9.13	15904	11.87	1.01	6	712	12228	9.13	15904	11.87	1.01
24	5	704	16197	12.09	16719	12.48	1.00	5	704	16197	12.09	16719	12.48	1.00
25	5	704	16197	12.09	17386	12.98	1.00	5	704	16197	12.09	17386	12.98	1.00
26	5	690	15973	11.92	17889	13.35	1.01	5	690	15973	11.92	17889	13.35	1.01
27	5	690	17039	12.72	18715	13.97	1.00	5	690	17039	12.72	18715	13.97	1.00
28	5	687	16208	12.10	19337	14.43	1.00	5	687	16208	12.10	19337	14.43	1.00
29	5	686	16551	12.36	20050	14.97	1.00	5	686	16551	12.36	20050	14.97	1.00

TABLE B.4: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n76_k8 and E_n76_k10 from the benchmark VRP library [63], without the capacity constraint.*

Problem instance	E_n101_k8								E_n101_k14							
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)		k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	
1	31	2080	1127	0.86	1304	1.00	1.00		31	2080	1127	0.86	1304	1.00	1.00	
2	24	1703	1703	1.31	1973	1.51	0.99		24	1703	1703	1.31	1973	1.51	0.99	
3	19	1471	2432	1.86	2629	2.02	0.99		19	1471	2432	1.86	2629	2.02	0.99	
4	16	1283	3019	2.32	3290	2.52	0.99		16	1283	3019	2.32	3290	2.52	0.99	
5	14	1207	3607	2.77	3906	2.99	1.00		14	1207	3607	2.77	3906	2.99	1.00	
6	13	1141	4169	3.20	4609	3.53	0.99		13	1141	4169	3.20	4609	3.53	0.99	
7	12	1108	4814	3.69	5265	4.04	0.99		12	1108	4814	3.69	5265	4.04	0.99	
8	11	1023	5250	4.03	5919	4.54	0.99		11	1023	5250	4.03	5919	4.54	0.99	
9	10	1020	6360	4.88	6577	5.04	0.99		10	1020	6360	4.88	6577	5.04	0.99	
10	10	987	6315	4.84	7247	5.56	0.99		10	987	6315	4.84	7247	5.56	0.99	
11	9	957	7355	5.64	7889	6.05	0.99		9	957	7355	5.64	7889	6.05	0.99	
12	9	944	7390	5.67	8551	6.56	0.99		9	944	7390	5.67	8551	6.56	0.99	
13	8	920	8508	6.52	9204	7.06	0.99		8	920	8508	6.52	9204	7.06	0.99	
14	8	932	9060	6.95	9855	7.56	0.99		8	932	9060	6.95	9855	7.56	0.99	
15	8	901	8888	6.82	10462	8.02	1.00		8	901	8888	6.82	10462	8.02	1.00	
16	8	891	9261	7.10	11097	8.51	1.00		8	891	9261	7.10	11097	8.51	1.00	
17	7	877	10811	8.29	11846	9.08	0.99		7	877	10811	8.29	11846	9.08	0.99	
18	7	887	11203	8.59	12503	9.59	0.99		7	887	11203	8.59	12503	9.59	0.99	
19	7	860	10990	8.43	13176	10.10	0.99		7	860	10990	8.43	13176	10.10	0.99	
20	7	841	11367	8.72	13838	10.61	0.99		7	841	11367	8.72	13838	10.61	0.99	
21	6	817	13779	10.57	14430	11.07	0.99		6	817	13779	10.57	14430	11.07	0.99	
22	6	819	13905	10.66	15104	11.58	0.99		6	819	13905	10.66	15104	11.58	0.99	
23	7	838	11361	8.71	15720	12.05	1.00		7	838	11361	8.71	15720	12.05	1.00	
24	6	818	15080	11.56	16313	12.51	1.00		6	818	15080	11.56	16313	12.51	1.00	
25	7	834	11849	9.09	17067	13.09	0.99		7	834	11849	9.09	17067	13.09	0.99	
26	6	820	14875	11.41	17781	13.63	0.99		6	820	14875	11.41	17781	13.63	0.99	
27	6	804	14981	11.49	18441	14.14	0.99		6	804	14981	11.49	18441	14.14	0.99	
28	6	796	14852	11.39	18746	14.37	1.01		6	796	14852	11.39	18746	14.37	1.01	
29	6	793	14596	11.19	19697	15.10	0.99		6	793	14596	11.19	19697	15.10	0.99	

TABLE B.5: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n101_k8 and E_n101_k14 from the benchmark VRP library [63], without the capacity constraint.*

Problem instance	E_n121_k7							E_n151_k12						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	37	5366	1529	0.77	1987	1.00	1.00	47	2988	1163	0.88	1317	1.00	1.00
2	27	4149	2621	1.32	2983	1.50	1.00	34	2282	1823	1.38	1975	1.50	1.00
3	22	3160	3533	1.78	3978	2.00	1.00	27	1898	2442	1.85	2632	2.00	1.00
4	18	2663	4460	2.25	4973	2.50	1.00	23	1730	3077	2.34	3295	2.50	1.00
5	15	2326	5454	2.75	5966	3.00	1.00	20	1572	3786	2.87	3942	2.99	1.00
6	13	2078	6548	3.30	6962	3.50	1.00	19	1485	3995	3.03	4608	3.50	1.00
7	12	1908	7240	3.64	7945	4.00	1.00	16	1357	5054	3.84	5272	4.00	1.00
8	11	1738	8059	4.06	8949	4.50	1.00	15	1305	5532	4.20	5921	4.50	1.00
9	10	1577	8691	4.37	9944	5.01	1.00	14	1294	6369	4.84	6564	4.98	1.00
10	9	1561	10468	5.27	10938	5.51	1.00	13	1226	6885	5.23	7243	5.50	1.00
11	9	1469	10450	5.26	11933	6.01	1.00	12	1131	7437	5.65	7889	5.99	1.00
12	8	1407	12521	6.30	12925	6.51	1.00	13	1159	6794	5.16	8566	6.50	1.00
13	8	1356	12464	6.27	13928	7.01	1.00	11	1127	8822	6.70	9201	6.99	1.00
14	9	1316	11493	5.79	14922	7.51	1.00	11	1091	8758	6.65	9851	7.48	1.00
15	7	1267	14455	7.28	15898	8.00	1.00	10	1066	10013	7.60	10516	7.99	1.00
16	7	1265	14537	7.32	16912	8.51	1.00	10	1060	10286	7.81	11166	8.48	1.00
17	7	1232	15333	7.72	17904	9.01	1.00	10	1053	10135	7.70	11826	8.98	1.00
18	6	1164	18594	9.36	18899	9.51	1.00	10	1062	10854	8.24	12507	9.50	1.00
19	6	1103	17894	9.01	19864	10.00	1.00	9	1023	12119	9.20	13175	10.01	1.00
20	6	1010	17665	8.89	20882	10.51	1.00	9	1021	11976	9.09	13772	10.46	1.00
21	6	1021	17753	8.94	21883	11.02	1.00	9	984	11663	8.86	14471	10.99	1.00
22	6	991	18509	9.32	22839	11.50	1.00	9	1016	12507	9.50	15112	11.48	1.00
23	5	1012	23457	11.81	23874	12.02	1.00	9	1021	13170	10.00	15795	11.99	1.00
24	5	1003	24327	12.25	24845	12.51	1.00	8	1009	15413	11.70	16474	12.51	1.00
25	5	960	22118	11.13	25832	13.00	1.00	8	986	15237	11.57	17077	12.97	1.00
26	5	989	23086	11.62	26842	13.51	1.00	8	990	15528	11.79	17791	13.51	1.00
27	5	982	23390	11.77	27853	14.02	1.00	8	977	15456	11.74	18451	14.01	1.00
28	5	980	23454	11.81	28835	14.51	1.00	8	978	16368	12.43	19014	14.44	1.00
29	5	975	22995	11.58	29822	15.01	1.00	8	984	16952	12.87	19725	14.98	1.00

TABLE B.6: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n121_k7 and E_n151_k12 from the benchmark VRP library [63], without the capacity constraint.*

Problem instance	E_n200_k16						
Iteration	k	d	Avg R	RF (avg)	Max R	RF (max)	% R (max)
1	64	3917	1179	0.88	1334	1.00	1.00
2	45	2987	1883	1.41	2001	1.50	1.00
3	36	2424	2477	1.86	2667	2.00	1.00
4	30	2088	3075	2.30	3314	2.48	1.00
5	26	1922	3715	2.79	4002	3.00	1.00
6	24	1819	4199	3.15	4664	3.50	1.00
7	22	1743	4944	3.71	5332	4.00	1.00
8	20	1589	5389	4.04	5974	4.48	1.00
9	18	1508	6139	4.60	6670	5.00	1.00
10	18	1532	6565	4.92	7315	5.48	1.00
11	16	1406	7489	5.61	7991	5.99	1.00
12	16	1389	7653	5.74	8667	6.50	1.00
13	15	1352	8332	6.25	9297	6.97	1.00
14	15	1354	8657	6.49	10001	7.50	1.00
15	14	1348	9539	7.15	10670	8.00	1.00
16	13	1302	10446	7.83	11330	8.49	1.00
17	13	1343	11438	8.57	11993	8.99	1.00
18	13	1303	11281	8.46	12654	9.49	1.00
19	13	1302	11675	8.75	13310	9.98	1.00
20	12	1248	12254	9.19	13990	10.49	1.00
21	11	1245	14065	10.54	14674	11.00	1.00
22	12	1244	13073	9.80	15324	11.49	1.00
23	11	1200	13965	10.47	15991	11.99	1.00
24	11	1181	14305	10.72	16663	12.49	1.00
25	11	1195	14314	10.73	17340	13.00	1.00
26	10	1172	16162	12.12	17988	13.48	1.00
27	11	1183	13939	10.45	18671	14.00	1.00
28	10	1154	16518	12.38	19307	14.47	1.00
29	10	1132	16470	12.35	19993	14.99	1.00

TABLE B.7: *The results of the parameter variation performed on the RCTVRP model for problem instance E_n200_k16 from the benchmark VRP library [63], without the capacity constraint.*